Tampereen yliopisto

Daniel Söderdahl

NIKKELIKATODIEN LAJITTELU KONENÄKÖÄ HYÖDYNTÄEN JA LAJITTELULINJASTON ESISUUNNITTELU

Diplomityö
Tekniikan ja luonnontieteiden tiedekunta
Tarkastaja: Esa Rahtu
Tarkastaja: Heikki Huttunen
Lokakuu 2020

TIIVISTELMÄ

Daniel Söderdahl: Nikkelikatodien lajittelu konenäköä hyödyntäen ja lajittelulinjaston esisuunnittelu
Diplomityö
Tampereen yliopisto
Automaatiotekniikka
Lokakuu 2020

Erilaisia lajittelu- ja luokittelutehtäviä on paljon ja monella alalla. Usein lajittelua tehdään manuaalisesti, mutta monessa paikassa on jo siirrytty konenäön avulla tehtäviin lajitteluihin. Manuaalinen lajittelu voi olla epäjohdonmukaista, aikaa vievää ja pitkästyttävää työtä.

Tämä työ on osa Tampereen yliopiston automaatiotekniikan diplomi-insinöörityötkintoa ja sen toimeksiantaja on Norilsk Nickel Harjavalta Oy. Työn tarkoitus on selvittää, soveltuuko konenäkö nikkelikatodien lajitteluun. Lisäksi työssä on esisuunniteltu tehtaalle mahdollisesti tulevaa lajittelulinjastoa.

Nikkelikatodien lajittelu tehdään tällä hetkellä manuaalisesti. Laitteistoa, jota lajittelussa käytetään, ei ole suunniteltu kyseiseen työhön. Tällä laitteistolla työ on riskialtista, sillä nikkelikatodit saattavat tippua siirrettäessä. Automaattisella lajittelulla saataisiin vapautettua työvoimaa ja tehtyä lajittelusta turvallisempaa sekä johdonmukaisempaa.

Työ jakaantuu kolmeen osaan. Teoriaosassa on käyty läpi konenäön menetelmiä, erityisesti konvoluutioneuroverkkoja. Kokeellisessa osassa konenäköä testattiin sekä kaupallisella ratkaisulla, että avoimen lähdekoodin kirjastojen avulla tehdyillä ohjelmoinneilla. Viimeisessä osassa suunniteltiin mahdollista lajittelulinjastoa.

Kaupallista ratkaisua testattiin niin, että nikkelikatodeja lajiteltiin manuaalisesti ja joka katodista otettiin kuva. Aluksi kuvia opetettiin mallille, jonka jälkeen malli osasi arvioida laatuja. Samoja kuvia käytettiin myös vertailuksi tehdyissä ohjelmoinneissa. Kaupallisella ratkaisulla lajittelun tarkkuudet olivat yli 90 % ja vertailuksi tehdyissä ohjelmoinneissakin päästiin parhaimmillaan 90 %. Konenäkö soveltuu siis nikkelikatodien lajitteluun hyvin. Parempia tuloksia olisi todennäköisesti saatu, jos konenäkömallien parantamista ja datan keräämistä olisi jatkettu.

Lajittelulinjaston suunnittelussa hahmoteltiin neljän erilaisen ratkaisun mallia. Suunnittelussa tärkeimpänä huomiona oli valaistuksen merkitys. Testien aikana valaistuksen muutokset tuottivat ongelmia, joten automaattisella linjastolla kuvaaminen olisi hyvä suorittaa sille suunnitellussa valaistussa huoneessa tai kopissa ongelmien välttämiseksi. Toisena tärkeänä asiana esiin nousi yksittäisen katodin sekä lajiteltujen nippujen painojen määrittäminen. Yksittäisen katodin paino tarvitaan, jotta joukosta löydetään liian kevyet tai painavat katodit. Lajiteltujen nippujen painot tarvitaan, jotta tiedetään, milloin nippu tulee valmiiksi. Tästä syystä linjastolla olisi oltava vaakoja, jotta nämä saadaan punnittua.

Avainsanat: Lajittelu, konenäkö, konvoluutioneuroverkko, koneoppiminen

ABSTRACT

Daniel Söderdahl: Sorting nickel cathodes using machine vision and pre-designing the sorting line
Master of Science thesis
Tampere University
Automation engineering
October 2020

There are different sorting and classification tasks in many fields. Sorting is often done manually, but in many places, sorting is nowadays done with machine vision. Manual sorting can be an inconsistent, time consuming and tedious task.

This thesis is part of Tampere University's Master of Science in Automation engineering-degree and it has been commissioned by Norilsk Nickel Harjavalta Oy. The purpose of this thesis is to determine is machine vision suitable for sorting nickel cathodes. In addition, a sorting line that may come to the factory is predesigned.

Nickel cathodes are currently sorted manually. The equipment used for sorting is not designed for that task. With that equipment, sorting is risky because nickel cathodes might drop when moved. Automatic sorting would release labor and it could make sorting safer and more consistent.

Thesis is divided into three parts. In theory section machine vision methods, especially convolution neural networks are explained. In experimental section machine vision was tested with commercial solution and programming with open source libraries. In the last section possible sorting line was predesigned.

The commercial solution was tested by manually sorting nickel cathodes and taking images of each cathode. The images were taught to the model and after that the model was able to evaluate the qualities. The same images were also used in the programming that was made for comparison. With the commercial solution, the sorting accuracies were over 90 % and with programming that was made for comparison the best results were 90 %. Machine vision is therefore suiting well for sorting nickel cathodes. Better results would have been achieved if the improvement of machine vision models and data collection had been continued.

In the designing of the sorting line, four different solutions were designed. The most important thing to consider in the design was the importance of lighting. During the tests, changes in lighting caused problems, so it would be a good idea that images in automatic line is taken in lighted room or chamber designed for it to avoid problems. Another important issue was the defining of the weights of a single cathode as well as sorted bundles. The weight of a single cathode is needed to find cathodes that are too light or heavy. Bundle weights are needed to know when the bundle is ready. For this reason, the line should have scales to weigh these.

Keywords: Sorting, machine vision, convolution neural network, machine learning

ALKUSANAT

Tämä diplomityö on tehty yhteistyössä Norilsk Nickel Harjavalta Oy:n kanssa. Haluan kiittää kaikkia NNH:n puolelta mukana olleita henkilöitä, jotka ovat auttaneet ja mahdollistaneet työn tekemisen. Erityiskiitos työn ohjaajalle Tuomo Laukkaselle. Häneltä sain paljon apua varsinkin kirjoittamisprosessin aikana.

Haluan kiittää myös Labra.AI yritystä, jolta konenäkölaitteisto kokeelliseen osuuteen vuokrattiin. Heiltä sain paljon apua kokeellisen osuuden aikana ongelmien ratkaisemiseen. He pitivät ohjelmiston ajan tasalla ja päivitettynä.

Koulun puolelta työtä ohjasi Esa Rahtu. Hänelle kiitos kaikesta palautteesta ja korjausehdotuksista. Häneltä sain hyödyllisiä vinkkejä sekä apua tätä työtä varten tehtyihin ohjelmointeihin.

Kiitokset myös perheelleni ja ystävilleni kaikesta tuesta opiskelujen aikana.

Tampereella, 27.10.2020

Daniel Söderdahl

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA SUOMENNOKSET

ANN	engl. Artificial neural network
CNN	engl. Convolutional neural network, konvoluutioneuroverkko
KNN	engl. K-nearest neighbor
NNH	Norilsk Nickel Harjavalta Oy
PCA	engl. Principal component analysis, pääkomponenttianalyysi
RCF	engl. Random cut forest
ReLU	engl. Rectified linear unit
R-CNN	engl. Regions with CNN features
SVM	engl. Support vector machine
YOLO	engl. You only look once

Alinäytteistys	Subsampling
Alueen ehdotus	Region proposal
Ankkurilaatikko	Anchor box
Askel	Stride
Epok	Epoch
Erä	Batch
Etuala	Foreground
Kehys	Frame
Kerros	Layer
Konenäkö	Machine vision
Koneoppiminen	Machine learning
Konvoluutioneuroverkko	Convolutional neural network
Kuvien käsittely	Image processing
Kynnysarvo	Threshold
Leikkauksen ja yhdisteen suhde	Intersect over union
Liukuikkuna	Sliding window
Luokittelija	Classifier
Monikerroksinen perseptroni	Multilayer perceptron
Neuroverkko	Neural network
Ohjaamaton oppiminen	Unsupervised learning
Ohjattu oppiminen	Supervised learning
Oppimisaste	Learning rate
Paino/tärkeys	Weight
Piirre	Feature
Pääkomponenttianalyysi	Principal component analysis
Rajausruutu	Bounding box
Ryhmittely	Clustering
Syväoppiminen	Deep learning
Syöte	Input
Tausta	Background
Tekoäly	Artificial intelligence
Tietokonenäkö	Computer vision
Tulos/ulostulo	Output
Tunniste	Label
Tunnistin	Detector
Täysin yhdistetty-kerros	Fully connected layer
Täyte	Padding
Varjojen tunnistus	Detect shadows
Ylinäytteistys	Upsampling

1. JOHDANTO

Teollisuudessa, maataloudessa sekä monella muulla alalla on paljon erilaisia lajittelu- ja luokittelutehtäviä, kuten esimerkiksi tuotteiden hylkäämistä, laadun erottelua, vikojen tunnistusta tai eri tuotteiden erottelua. Usein lajittelua tehdään manuaalisesti. Manuaalinen lajittelu on kuitenkin epäjohdonmukaista, aikaa vievää, vaihtelevaa ja subjektiivista. Manuaalinen lajittelu sitoo työvoimaa ja näin ollen se on myös usein kallista. Lisäksi se on ihmiselle usein pitkästyttävää ja ympäristö sekä sen tapahtumat saattavat vaikuttaa lajitteluun. Näistä syistä manuaalista lajittelua on alettu korvaamaan erilaisilla automaattisilla ratkaisuilla. [1, 2]

Varsinkin maataloudessa ja teollisuudessa manuaalisen lajittelun tilalle on kehitelty konenäköön sekä neuroverkkoihin perustuvia lajitteluratkaisuja. Konenäön avulla on lajiteltu esimerkiksi erilaisia hedelmiä ja vihanneksia. Lajittelua on tehty muun muassa värin, koon ja muodon perusteella sekä erilaisten vikojen perusteella. [1, 2] Konenäön käyttö on helpottunut konvoluutioneuroverkkojen (CNN) myötä, sillä niille syötettävät kuvat eivät vaadi niin paljon esikäsittelyä kuin muut luokittelumenetelmät [3].

Luokittelumenetelmissä hyödynnetään usein koneoppimista. Siinä luokittelumallille ei anneta tarkkoja ohjeita, vaan malli oppii luokittelemaan kuvioiden ja päättelyn avulla. Koneoppimismenetelmät rakentavat mallin annetun datan perusteella, josta voidaan tehdä ennustuksia ja päätöksiä. [4]

Konenäöllä suoritettaviin lajittelu- ja luokittelutehtäviin kuuluu olennaisesti kuvan esikäsittely, kuvan segmentointi, kappaleen tunnistaminen ja kappaleen luokittelu. Segmentointimenetelmiin kuuluvat esimerkiksi *thresholding, background substraction* sekä U-Net, tunnistamismenetelmiin *You only look once* (YOLO) sekä *regions with CNN features* (R-CNN) ja luokittelumenetelmiin pääkomponenttianalyysi (PCA) sekä CNN. Konvoluutioneuroverkot ovat myös pohjana monelle menetelmälle, kuten R-CNN. CNN:llä on paljon muitakin käyttökohteita luokittelun lisäksi, esimerkiksi kasvojen tunnistuksessa.

Tämän työn tarkoituksena on selvittää konenäön soveltuvuus nikkelikatodien lajitteluun. Nikkelikatodien laatuvaatimukset ovat kasvaneet markkinoilla ja nikkelikatodien lajittelua on alettu tekemään manuaalisesti. Työssä on esisuunniteltu mahdollista automaattista lajittelulinjastoa. Tutkimuskysymyksinä on: soveltuuko konenäkö nikkelikatodien lajitteluun ja mitä asioita on huomioitava lajittelulinjaston suunnittelussa? Konenäön soveltuvuuskysymykseen liittyy myös kysymykset: paljonko luokittelumallia on testattava, jotta

saadaan riittävä tarkkuus ja mikä on riittävä tarkkuus, jotta voidaan todeta konenäön soveltuvuus?

Tutkimuksessa on ensin testattu kaupallista luokittelumallia, jolla saadaan otettua nikkelikatodeista kuvat, segmentoitua ne, opetettua kuvat luokittelumallille ja testattua uusia kuvia. Tämän jälkeen on testattu avoimen lähdekoodin avulla konvoluutioneuroverkkoja luokittelussa. Linjaston suunnittelussa apuna on käytetty luokittelutesteissä havaittuja asioita. Myös työntekijöitä on haastateltu linjaston suunnittelussa huomioitavista asioista.

Ensimmäisessä luvussa on käyty läpi työn taustoja. Seuraavassa luvussa on kerrottu teoriaa konenäöstä sekä käyty läpi erilaisia tutkimuksia konenäön käytöstä lajitteluissa. Teoriassa on keskitytty eniten konvoluutioneuroverkkoihin. Teorian jälkeen on kokeellinen osuus, jossa kerrotaan tehdyistä testeistä ja ohjelmoinnista. Kokeellisen osuuden yhteydessä suoritettiin myös leikattujen nikkelipalojen lajittelutesti. Ohjelmoinnissa on keskitytty konvoluutioneuroverkkoratkaisuihin sekä pieniin ja yksinkertaisiin malleihin. Kokeellisen osuuden jälkeen on kaupallisen ratkaisun ja avoimen lähdekoodin avulla tehtyjen ohjelmointien tulosten käsittelyä. Lopussa on vertailtu niistä saatuja tuloksia sekä tehty päätelmiä. Tuloksien ja päätelmien jälkeen on tehty esisuunnittelua lajittelulinjastoa varten. Viimeisenä tulokset ja huomiot on koottu yhteen.

2. TYÖN TAUSTA

Tässä luvussa kerrotaan ensin Norilsk Nickel Harjavalta Oy:stä. Sen jälkeen on esitetty tutkimusongelma sekä nykyinen manuaalinen lajittelupiste. Lopuksi on esitelty vielä lyhyesti Labra.AI yritys, jolta konenäkölaitteisto vuokrattiin.

2.1 Norilsk Nickel Harjavalta Oy

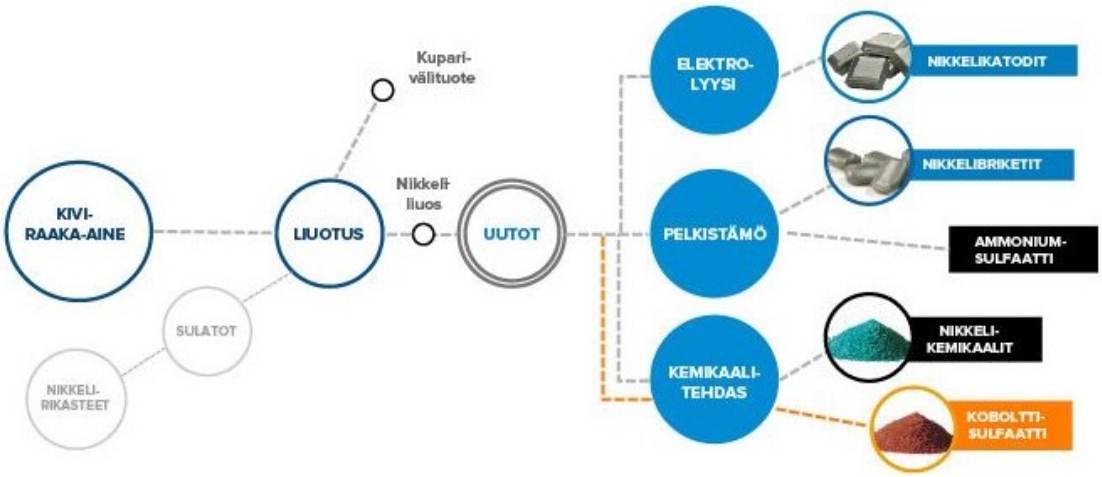
Norilsk Nickel Harjavalta Oy (NNH) on osa kansainvälistä Nornickel-konsernia, joka on maailman suurin nikkelin ja palladiumin tuottaja. Nornickel tuottaa 14 % koko maailman nikkelistä ja lisäksi merkittäviä määriä platinaa, rhodiumia, kobolttia ja kuparia. Muita tuotteita ovat kulta, hopea, iridium, seleeni, rutenium ja telluuri. [5]

NNH sijaitsee Harjavallan suurteollisuuspuistossa, jossa toimii noin 20 yritystä. NNH ja Boliden Harjavalta Oy ovat suurteollisuuspuiston isoimmat yritykset. [6] NNH:lla työskentelee noin 300 henkilöä ja sen liikevaihto vuonna 2019 oli yli miljardi euroa [7]. NNH:n vuosituotantokapasiteetti on 66 000 tonnia nikkeliä [5]. NNH tuottaa koko konsernin nikkelituotannosta 28 %. NNH koostuu neljästä osastosta, jotka ovat liuottamo, uutto-pelkistämö, elektrolyysi ja kemikaalitehdas. [8]

NNH:n päätuotteita ovat nikkelikatodit, nikkelibriketit ja nikkelikemikaalit. Nikkelikatodien osuus Harjavallan kokonaistuotannosta on 29 % ja niiden nikkelipitoisuus on 99,9 %. Nikkelibrikettien osuus on 58 % ja ne ovat 99,8 % nikkeliä. Nikkelikemikaalien osuus on 13 %. Nikkelikemikaaleja ovat nikkelisulfaatti, nikkelihydroksidi ja nikkelihydroksikarbonaatti. Lisäksi NNH:lta tulee sivutuotteina kobolttisulfaattia ja ammoniumsulfaattia. [8]

Tuotantoprosessi alkaa liuottamolta, jossa raaka-aineet liuotetaan kiintoaineesta vesiliuokseen. Tässä prosessivaiheessa erotetaan myös epäpuhtaudet, kuten rauta ja kupari. Liuottamolta tulee myös välituotteena kuparisakkaa, joka myydään eteenpäin kuparin jatkojalostukseen. [8]

Liuottamolta nikkeliraakaliuos menee uuttoihin, jossa liuoksesta puhdistetaan lisää epäpuhtauksia, kuten koboltti ja kalsium. Kalsiumin ja koboltin lisäksi liuoksesta uutetaan rautaa, sinkkiä, kuparia ja mangaania. Uutoista puhdistettu nikkeliliuos johdatetaan elektrolyysiin ja pelkistämöön jatkokäsittelyyn. Kobolttiraakaliuosta puhdistetaan lisää ennen sen siirtoa kemikaalitehtaalle. [8]



Kuva 1. *NNH:n prosessikuvaus ja tuotteet. Työssä lajiteltavat nikkelikatodit tulevat elektrolyysistä [8].*

Pelkistämössä nikkeliliuos pelkistetään vedyn avulla nikkelipulveriksi, joka erotetaan liuoksesta. Tämän jälkeen nikkelipulveri briketoidaan eli puristetaan kappaleeksi koneellisesti. Briketoinnin jälkeen briketit typpisintrataan, jonka tarkoituksena on lujittaa briketejä. Tässä prosessin vaiheessa syntyy ammoniumsulfaattiliuosta, josta erotetaan liuokseen jäänyt nikkeli, joka palautetaan prosessiin. Ammoniumsulfaattiliuos kiteytetään ja myydään lannoitekäyttöön. [8]

Elektrolyysissä tuotetaan nikkelikatodeja *electrowinning*-menetelmällä. Menetelmässä tasavirtaa syötetään liukenemattoman lyijyanodin kautta elektrolyyttiin ja nikkelikatodille, johon nikkeli siirtyy nikkeliliuoksesta sähkövirran avulla. [8] Siemenlevyt, jotka laitetaan elektrolyysialtaisiin, ovat noin 8-12 kiloisia. Ne ovat elektrolyysialtaissa kasvamassa noin 6-7 vuorokautta, jonka jälkeen ne ovat 60-80 kiloisia. Yksi nikkelikatodi on noin neliömetrin kokoinen. Valmiit katodit menevät leikkaamolle, jossa ne tuotteistetaan, joko kokolevyinä tai leikattuna ja pakattuna. Tuotteistaminen tarkoittaa, että katodit on saatettu lopputuotteeksi, jolloin se on myyntikunnossa.

Uutosta tuleva erittäin puhdas nikkelisulfaattiliuos menee kemikaalitehtaalle, jossa siitä valmistetaan epäorgaanisia suoloja. Kemikaalitehtaalla on omat tuotantolinjat nikkelisulfaatille, nikkelihydroksidille ja nikkelihydroksikarbonaatille. Näiden lisäksi kobolttisulfaattiliuoksesta kiteytetään kobolttisulfaattikiteitä. [8] Yksinkertaistettu kuvaus NNH:n tuotantoprosessista on esitetty Kuvassa 1.

2.2 Ongelman asettelu

NNH:n elektrolyysistä tulevat katodit ovat pinnanlaadultaan erilaisia. Katodien luokista käytetään termejä priima, standardi ja romu. Priimalaatu on pinnaltaan parhaimman näköistä. Standardilaatu on yleisin laatu. Priimalaadun nikkelikatodi on halutumpaa kuin

standardilaadun katodi, koska sen pinta on tasaisempaa kuin standardilaadun katodeilla, vaikka ne ovat kemiallisesti yhtä hyviä. Priimalaadun katodilla on myös parempi kate, koska se menee paremmin maksaville asiakkaille pinnoitukseen [9]. Standardi ja romulaadun katodit taas menevät osittain ruostumattoman teräksen tekoon [9].

Katodit tulevat leikkaamolle noin 24 kappaleen nipuissa, joita tulee päivässä 30-40. Elektrolyysissä yhdestä altaasta tulee 48 katodia ja kun nämä laitetaan kahteen nippuun, saadaan hyvän painoisia sekä kokoisia nippuja. Nipuissa voi olla vain muutama priimalaadun katodi tai muutamia huonompia katodeja. Huonot katodit estävät koko nipun tuotteistamisen priimalaadun tuotteeksi. Elektrolyysissä pyritään poistamaan suurin osa romukatodeista ennen niputtamista, mutta silti niitä jää joihinkin nippuihin. Näistä syistä katodiniput pitää lajitella. Näin saadaan maksimoitua priimalaadun saanti ja poistettua nipuista romukatodit, joita ei voida leikata. Priimalaadun katoditkaan eivät mene lajittelun ansiosta turhaan leikkaukseen.

Tällä hetkellä lajittelu suoritetaan manuaalisesti laitteistolla, jota ei ole suunniteltu eikä tarkoitettu tähän käyttötarkoitukseen. Työ ei ole ergonomista ja lajittelu sitoo joka vuorossa yhden työntekijän lajitteluun. Jos käytössä olisi automaattinen linjasto, saisi työntekijän vapautettua muihin työtehtäviin. Koska lajittelijoita on monta, on lajittelussa ja priimalaadun saannissa tällä hetkellä eroja. Toiset ovat tarkempia lajittelussa, jolloin priimalaatua saattaa tulla vähemmän, kuin jos joku muu olisi lajitellut. Automaattisen linjaston ja konenäön avulla laatu olisi tasaisempaa. Manuaaliselta lajittelulinjastolta puuttuu myös vaaka. Nippujen painot pitäisi tietää, jotta tiedettäisiin, milloin nippu on valmis. Tällä hetkellä vain arvioidaan nippuun oikea paino pinoamalla nippuun 23-25 katodia.

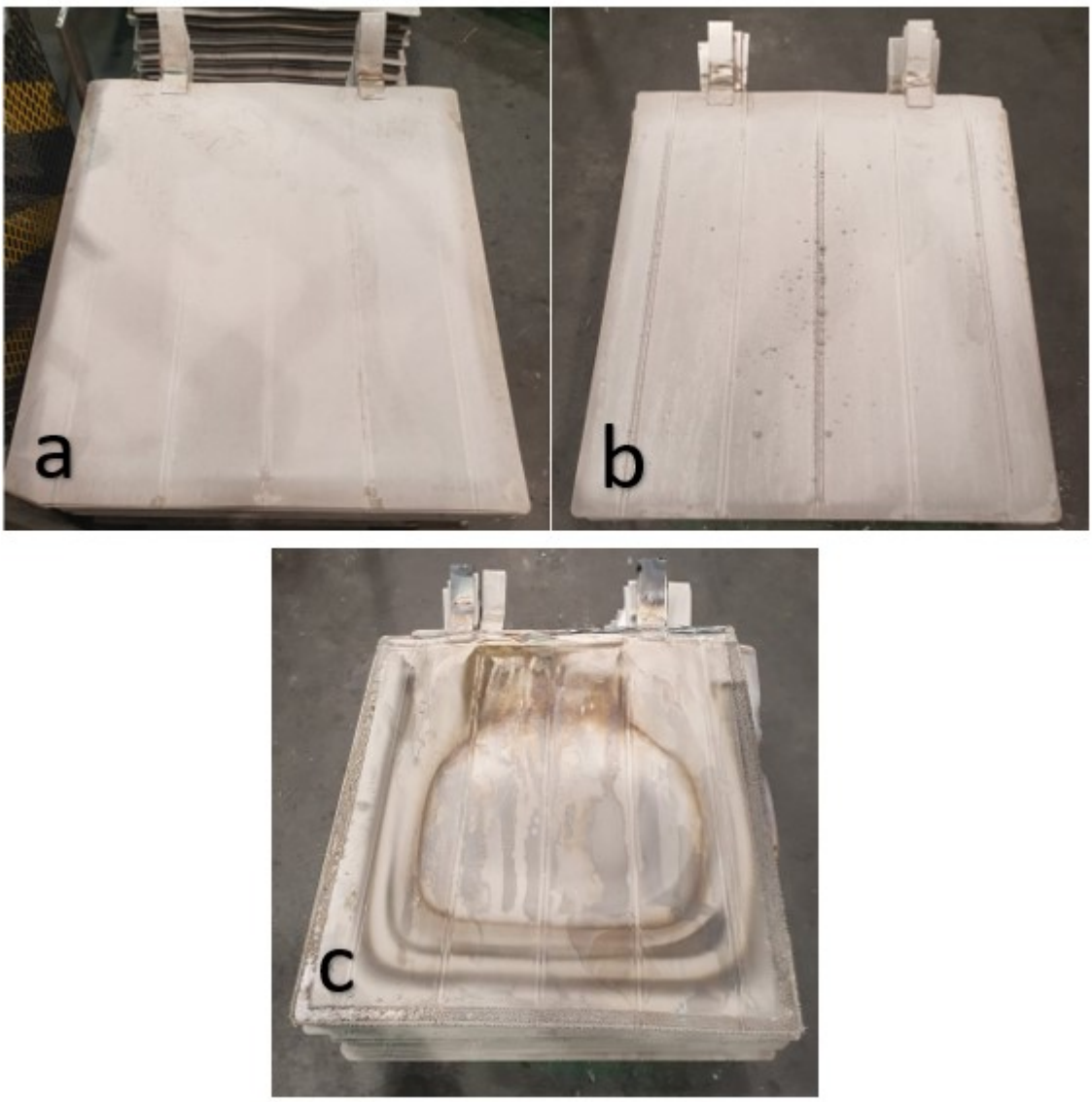
2.3 Manuaalinen lajittelupiste

Lajittelu suoritetaan NNH:n leikkaamolla imukuppinostimen avulla. Lajittelupiste koostuu kahdesta katodinipuille ja lajitteluun räätälöidystä metallipöydästä, imukuppinostimesta sekä trukkilavoista, joiden päälle lajiteltava nippu ja romunippu asetetaan. Kuvassa 2 lajittelupiste on esitetty tyhjänä sekä nippujen kanssa. Kuvassa 3 on esitetty lajittelussa käytetty imukuppinostin. Imukuppinostimen avulla nikkelikatodit saadaan siirrettyä nipusta toiseen.

Kuva 2. *Lajittelupiste tyhjänä vasemmalla ja nippujen kanssa oikealla.*

Kuva 3. *Lajittelussa käytetty imukuppinostin.*

Ensin lajiteltava nippu tuodaan trukilla lajittelupisteelle. Tämän jälkeen nippu lajitellaan kolmeen eri luokkaan: priimalaatuun, standardilaatuun ja romulaatuun. Priimalaatu on pinnanlaadultaan parhaimman näköistä nikkelikatodia. Priimalaadun katodin pitää olla sileäpintainen ja oikean värinen. Katodien pinnassa voi olla erilaisia pinnanlaatuvirheitä kuten värivirheitä, dendriittejä tai vetyreikiä. Värivirhe voi olla esimerkiksi kupariraita. Dendriitit ovat pisaramaisia kasvannaisia katodin pinnassa. Muodostuakseen dendriitit tarvitsevat jonkin hiukkasen katodin pinnalla, jonka ympärille nikkeli alkaa kasvamaan [9]. Dendriittejä on eri kokoisia riippuen ajasta, jolloin se on lähtenyt kasvamaan. Vetyreiät ovat kohtia levyn pinnassa, joihin ei ole kasvanut nikkeliä [9]. Ne saattavat johtua liiasta vedystä [9]. Priimalaadun katodiin voidaan sallia myös pieniä määriä pinnanlaatu



Kuva 4. *Nikkelikatodin laadut: priimalaatu (a), standardilaatu (b) ja romulaatu (c).*

virheitä ja siksi luokittelu onkin hankalaa. Jos virheitä on paljon, luokitellaan katodi standardilaaduksi. Romulaadun katodit ovat selkeästi palaneita, mustia tai niillä on jo todella paljon dendriittikasvustoa. Nikkelikatodin laadut on esitetty Kuvassa 4.

Niput ovat valmiita, kun niissä on katodeja noin 24 kappaletta. Valmis nippu määritellään sen painon mukaan, joka on noin 1 500 kg. Kun priimalaadun nippu tulee valmiiksi, se punnitaan, sidotaan vanteilla ja tuotteistetaan. Standardilaadun nipun valmistuttua, se viedään leikkaukseen sellaisenaan tai punnitaan, sidotaan vanteilla ja tuotteistetaan, kuten priimalaadunkin nippu. Myös romuniput punnitaan, sidotaan vanteilla ja tuotteistetaan.

Manuaalinen lajittelu sisältää myös riskejä. Katodien reunat tai kulmat voivat lajittelun aikana osua jalkoihin. Katodit saattavat olla teräviä ja niistä voi saada haavoja jalkoihin tai housut saattavat revetä. Huonosti trukkilavalle asetettu katodinippu voi myös kaatua

ja aiheuttaa suurta vahinkoa. Suurin ja todennäköisin riski on kuitenkin katodin tippuminen. Kun imukuppinostimella nostetaan katodia, se saattaa pudota. Yksi nikkelikatodi painaa noin 65 kiloa ja jos se tippuu esimerkiksi jalan tai säären päälle, jalka voi murtua. Nippujen siirrot tehdään trukilla, jolloin ihmiselle ei aiheudu riskiä nippujen kaatumisesta. Muuten riskit on huomioitu työssä vain huolellisuudella ja varovaisuudella.

2.4 Labra.AI

Diplomityön kokeellisen osan suorittamiseen NNH vuokrasi konenäkölaitteiston ja ohjelman Labra.AI yritykseltä. Labra.AI on konsultointi yritys, joka on erikoistunut tekoälyyn. Se on espoolainen yritys, joka on perustettu 2018. [10]

Labra.AI:n Vision on moderni, syväoppimisen mahdollistava konenäköjärjestelmä. Sitä voi käyttää laadun varmistamiseen, mittaamisen ja datan keräämiseen. Vision käyttää neuroverkkoja piirteiden *(feature)* oppimiseen sille näytetyistä kappaleista. Ohjelmisto opettaa neuroverkon vertaamalla esimerkkikuvia, jotka käyttäjä on luokitellut eri luokkiin. [10]

3. KONENÄKÖ LAJITTELUSSA

Tässä luvussa kerrotaan ensin yleisesti konenäöstä, jonka jälkeen siirrytään kappaleen tunnistamiseen sekä sen menetelmiin. Kappaleen tunnistamisen jälkeen selostetaan kuvan segmentoimista. Segmentoimisen jälkeen kerrotaan luokittelumenetelmistä ja niiden jälkeen laajemmin konvoluutioneuroverkoista. Konvoluutioneuroverkkojen jälkeen esitellään datan käsittelyä. Lopuksi käydään läpi vielä erilaisia konenäön avulla tehtyjä lajittelusovelluksia.

Ihmisen näkö on todella vaikuttava. Se selviytyy hyvin intensiteetin, värin, muodon ja tekstuurin vaihteluista. Ihminen pystyy näkemään jo pienenkin värimuutoksen, joka on vaikea tunnistaa koneella. Ihminen pystyy tunnistamaan tutut kasvot väkijoukosta. Ihmisen näön avulla teollisuudessa saadaan suoritettua esimerkiksi tarkastamisia, arvosteluja, lajittelua, laskemista ja monia muita tehtäviä. Teollisuudessa ihmisen näkökyky on monessa paikassa hyödyllinen, mutta sillä on myös rajansa, joita ovat esimerkiksi väsymys, epämukavuus, sairaudet, häiriöt ja tylsyys. [11]

Monet tuotantoprosessien tehtävät ovat niin nopeita, että ihminen ei pysy niiden vauhdissa mukana. Tehtävien tylsyys tuo epäluotettavuutta ja heikentää tarkkuutta. Jotkut tehtävät vaativat tarkat mitat ja dimensiot, joita ihminen ei pysty silmämääräisesti määrittämään. Jotkin tehtävät voivat olla ihmiselle vaarallisia ympäristön tai käsiteltävien kappaleiden takia. Näistä syistä on pitkään yritetty kehittää konetta, joka ”näkee”. [11]

Termit konenäkö *(machine vision)*, tietokonenäkö *(computer vision)* ja koneoppiminen *(machine learning)* liittyvät konenäköön sekä sen käyttöön lajittelussa. Konenäkö ja tietokonenäkö termit voidaan ajatella monesti synonyymeinä, mutta niille on myös erilaisia määritelmiä, joita on seuraavaksi mainittu muutama.

Tietokonenäöllä tarkoitetaan automaattiseen kuvan tai videon ottamista sekä niiden käsittelyä ja analysointia. Konenäöllä taas tarkoitetaan tietokonenäön käyttöä teollisissa ympäristöissä. Konenäkö käsittää siis koko järjestelmän, jota tehtävässä tarvitaan. Jos käsitellään vain tietokoneella olevia tiedostoja, tehtävästä selvitään pelkällä tietokonenäöllä. Mutta jos tarvitaan reaalimaailmasta esimerkiksi kuva kolikosta, tarvitaan konenäköjärjestelmää. [11, 12]

Konenäöllä tarkoitetaan tiedettä ja teknologiaa, jonka avulla koneet sekä saavat että käyttävät visuaalista informaatiota teollisuudessa ja tuotannossa. Konenäköä hyödynnetään lajitteluissa paljon, koska sillä on monia etuja manuaaliseen lajitteluun verrattuna. Ihmisen tekemä manuaalinen lajittelu on usein pitkästyttävää työtä. Tarkkuus saattaa

vaihdella tekijöiden välillä ja myös väsymystila vaikuttaa tulokseen. Manuaalinen lajittelu saattaa myös hidastaa prosessin etenemistä. [13, 14]

Ihminen ei välttämättä pysy liukuhihnan vauhdissa esimerkiksi linjastoilla, joissa tavara liikkuu nopeasti liukuhihnalla. Osa lajittelutehtävistä on myös sellaisia, jotka vaativat kokeneen henkilön tehtävään. Konenäön käyttäminen tällaisissa tehtävissä lisäisi tuottavuutta, koska sillä mahdollistetaan myös nopeampi luokittelu. Sen avulla saataisiin myös ihminen pois pitkästyttävistä töistä ja näin ollen saataisiin vähennettyä työvoimakustannuksia. Tyypillisiä konenäön komponentteja ovat kappaleen käsittelyjärjestelmät, kuten robotit, valaistus, optiset laitteet, kamera, kuvan ja datan käsittely-yksikkö sekä ohjelmisto. [13, 14]

Tietokonenäöllä pyritään ymmärtämään kuvia ja videoita. Sillä yritetään kuvata maailma, kuten ihminen sen näkee. Konenäön avulla pyritään ymmärtämään ja automatisoimaan tehtäviä, joita ihmisen näön avulla pystytään tekemään. Tehtävät koostuvat kuvan ottamisesta, kuvan käsittelystä ja tulkitsemisesta. Data voi olla esimerkiksi videosekvenssejä, näkymiä monesta kamerasta tai multi-dimensionaalista dataa. Tietokonenäön sovelluksia on todella paljon, muutamana esimerkkinä automaattiset laadun tarkastukset, tunnistamiset ja luokittelut, robotin kontrollointi, kappaleiden ja ympäristön mallintaminen sekä navigaatio. [15, 16]

Tyypillisiä tietokonenäön tehtäviä ovat tunnistukset, liikkeen analysoinnit, rekonstruktiot sekä kuvan parantaminen. Tunnistukseen kuuluvat esimerkiksi kappaleen tunnistaminen, tiettyjen piirteiden tunnistaminen ja kasvojen tunnistus. Liikkeen analysointiin kuuluu esimerkiksi pisteen tai kappaleen liikkeen seuraaminen. Monet tietokonenäköjärjestelmät koostuvat datan keruusta, esikäsittelymetodeista, piirteiden irrottamismetodeista, segmentointimetodeista, korkean tason käsittelymetodeista sekä päätöksenteosta. [15, 16]

Koneoppiminen on menetelmien ja mallien tutkimusala, jota tietokonejärjestelmät käyttävät suorittamaan tiettyjä tehtäviä ilman tarkkoja ohjeita, luottaen kuvioihin sekä päättelyyn. Koneoppimismenetelmät rakentavat matemaattisen mallin annetusta datasta, jonka perusteella pystytään tekemään ennustuksia ja päätöksiä. [4, 17]

Koneoppiminen jaetaan ohjattuun *(supervised)* ja ohjaamattomaan *(unsupervised)* oppimiseen. Ohjattuun oppimiseen käytetään oikeaa tietoa *(ground truth),* jolloin tiedetään mitä tulosten tulisi olla. Ohjatun oppimisen tehtävänä on siis oppia funktio, joka annettujen syötteiden sekä haluttujen tulosten avulla arvioi parhaan suhteen syötteiden ja tulosten välille annetun datan perusteella. Ohjaamattomassa oppimisessa halutut tulokset eivät ole tiedossa, eli sen tehtävänä on päätellä datan luonnollinen rakenne. [18]

Ohjattua oppimista käytetään yleensä luokittelussa tai regressiossa. Yleisiä menetelmiä ovat logistinen regressio, *naive bayes, support vector machines (SVM), artificial neural networks ja random forests*. Sekä regressiossa että luokittelussa on tavoitteena löytää suhde tai rakenne syötedatassa, joka mahdollistaa oikean tuloksen tuottamisen. [18]

Ohjaamattoman oppimisen yleisimmät käyttökohteet ovat *clustering, representation learning ja density estimation.* Tavoitteena jokaisessa on oppia datan ilmeinen rakenne ilman annettuja tuloksia. Yleisiä algoritmeja ovat *k-means clustering,* PCA ja *autoencoders.* [18]

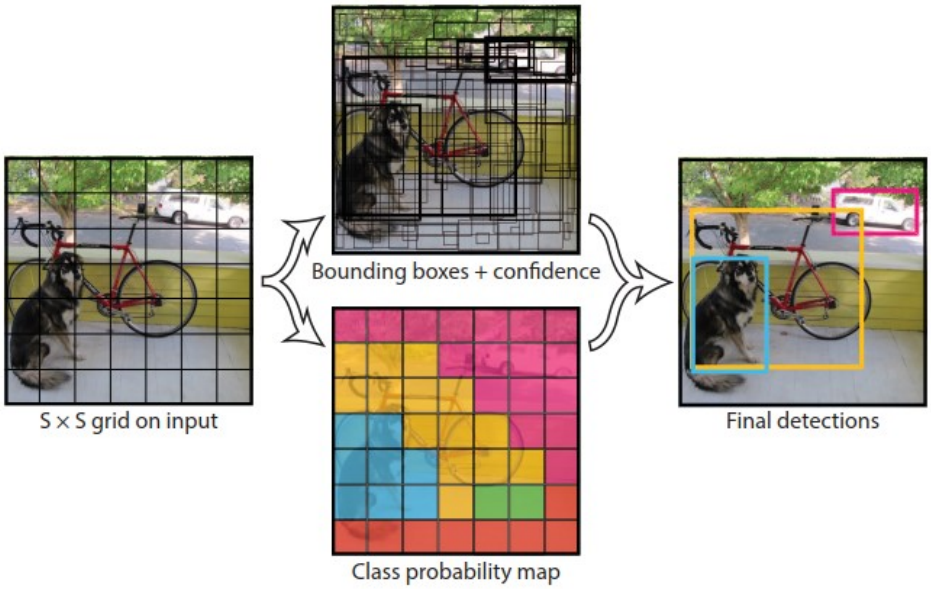
Esimerkkinä ohjatusta oppimisesta on hedelmäkori, jossa olevat hedelmät pitäisi tunnistaa. Ensiksi koneelle tarvitsee opettaa erilaisten hedelmien piirteet, kuten että omena on pyöreä ja punainen. Opettamisen jälkeen kone osaa tunnistaa omenat. [19]

Ohjaamattomasta oppimisesta esimerkkinä on kuva kissoista ja koirista, joita kone ei ole vielä nähnyt. Koneella ei ole tietoa koirien ja kissojen piirteistä, joten niitä ei voi luokitella kissoiksi ja koiriksi. Kone kuitenkin osaa luokitella ne samankaltaisuuksien, kuvioiden ja erojen perusteella kahteen osaan. [19]

3.1 Kappaleen tunnistaminen

Kappaleen tunnistamismenetelmät tunnistavat kuvasta kappaleet ja piirtävät rajausruutuja *(bounding box)* niiden ympärille. Ne siis paikantavat ja luokittelevat kappaleet. Tunnistusmenetelmiä on esimerkiksi YOLO ja R-CNN. [20]

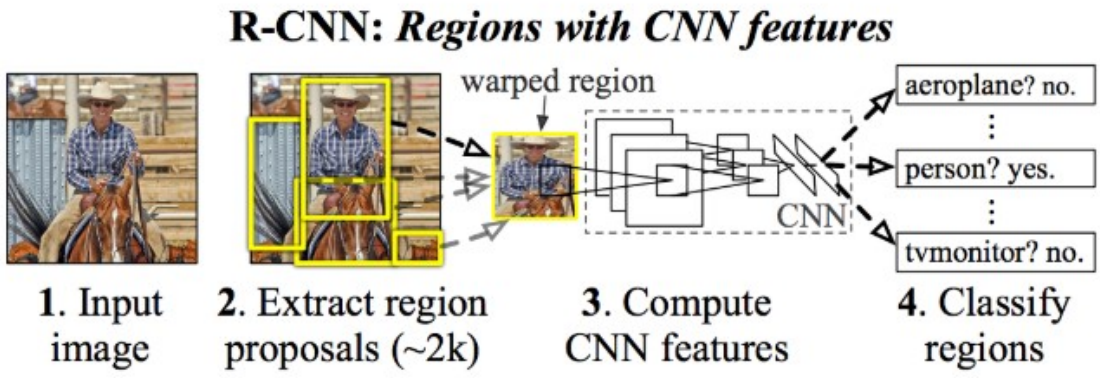
You Only Look Once (YOLO). Kuvaa katsoessa ihmiset tunnistavat ja tietävät heti missä kohdassa kuvaa kappaleet ja asiat sijaitsevat. Jos kone pystyisi samanlaiseen kappaleen tunnistamiseen, se mahdollistaisi monimutkaisempien tehtävien suorittamista konenäön avulla. Nykyiset tunnistus järjestelmät käyttävät luokittelumenetelmiä tunnistamiseen. Kappaleen tunnistamiseen nämä järjestelmät käyttävät kappaleen luokittelijaa ja arvioivat sitä eri paikoissa sekä skaaloissa testikuvassa. *Deformable parts models* käyttävät liukuikkuna *(sliding window)* lähestymistapaa, jossa luokittelija ajetaan koko kuvan läpi tasaisesti jaetuissa paikoissa. Alueen ehdotus *(region proposal)* menetelmät luovat ensin potentiaaliset rajausruudut kuvaan, jonka jälkeen luokittelija ajetaan näissä ruuduissa. Luokittelun jälkeen rajausruudut määritetään uudelleen ja niistä eliminoidaan ne, jotka esiintyvät monta kertaa. Lisäksi ruudut pisteytetään uudelleen muiden kuvassa olevien kappaleiden perusteella. Nämä menettelytavat ovat hitaita ja vaikeita optimoida. [21]



Kuva 5. *YOLOn malli [21].*

YOLO menetelmä muotoilee uudelleen kappaleen tunnistuksen yksittäiseksi regressio ongelmaksi. Se tekee kappaleen tunnistusta reaaliajassa ja menetelmässä mennään suoraan kuvan pikseleistä rajausruutujen koordinaatteihin ja luokkien todennäköisyyksiin. YOLO ”katsoo” kuvaa vain kerran, jonka perusteella se tunnistaa, mitkä kappaleet ovat kuvassa ja missä ne ovat. Se opettelee siis kokonaisilla kuvilla. YOLOssa yksittäinen konvoluutioverkko ennustaa samanaikaisesti monia rajausruutuja ja niiden luokkien todennäköisyyksiä. [21]

YOLO jakaa annetun kuvan S x S ruudukkoon. Jos kappaleen keskiosa osuu ruudukon soluun, kyseinen solu on vastuussa sen kappaleen tunnistamisesta. Jokainen solu ennustaa määrätyn määrän rajausruutuja ja niiden varmuus pisteet *(confidence points)*. Varmuus pisteet kertovat todennäköisyyden sille, että kyseisessä ruudussa on kappale ja kuinka tarkka sen rajausruutu on. Rajausruudun keskusta suhteessa ruudukon solun reunoihin on x, y. Ruudun leveys w ja korkeus h ennustetaan suhteessa koko kuvaan. Varmuuspiste on ennustetun ruudun ja minkä tahansa todellisen *(ground truth)* ruudun välisen leikkauksen ja yhdisteen välinen suhde *(intersect over union)*. Jokainen ruudukon solu ennustaa myös ehdollisten luokkien todennäköisyydet. Jokainen rajausruutu sisältää parametrit x, y, w, h ja varmuusennustukset. [21] YOLOn malli on esitetty Kuvassa 5.



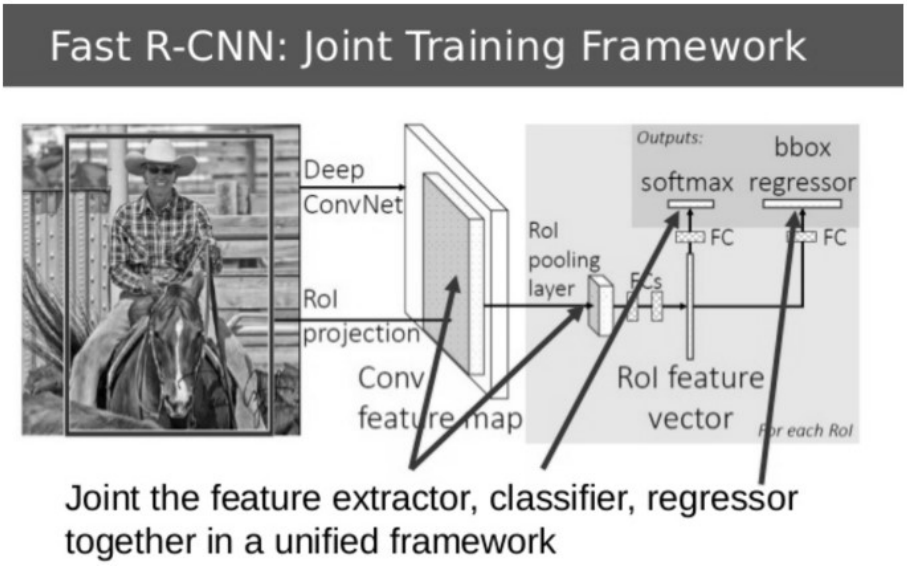
Kuva 6. *R-CNN:n malli [22].*

Regions with CNN features (R-CNN). Kappaleen tunnistuksessa ja segmentoimisessa käytetään myös R-CNN menetelmiä. Alkuperäisestä R-CNN sovelluksesta on tehty paranneltuja versioita. R-CNN menetelmiä ovat alkuperäinen R-CNN, Fast R-CNN, Faster R-CNN ja Mask R-CNN.

R-CNN pyrkii tuomaan luokittelun ja kappaleen tunnistuksen lähemmäs toisiaan. Se yhdistää alueen ehdotukset ja konvoluutioneuroverkot. R-CNN voidaan jakaa alueen ehdotusvaiheeseen ja luokitteluvaiheeseen. R-CNN koostuu kolmesta osasta ja sen malli on esitetty Kuvassa 6. Ensimmäinen luo alueen ehdotukset syötekuvalle. Toinen irrottaa jokaiselta ehdotetulta alueelta piirteet CNN:n avulla. Kolmas on kokoelma luokkakohtaisia SVM:iä, jotka luokittelevat alueet. [22, 23]

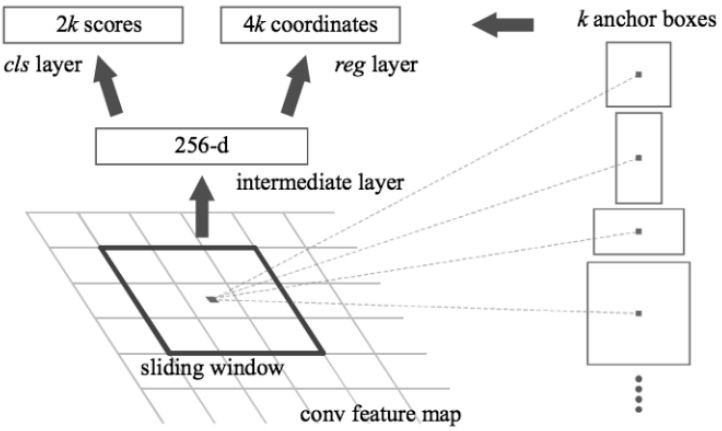
Alueen ehdotukset R-CNN:lle luodaan valikoiva haku *(selective search)* menetelmällä. Ehdotukset muutetaan tiettyyn kokoon, jotka CNN ottaa syötteenä (Kuvassa 6 *warped region*). CNN irrottaa piirrevektorin jokaiselle alueelle. Lopuksi SVM:ien avulla luokitellaan, onko ruuduissa kappaletta ja mitä mahdolliset kappaleet ovat. Lineaarisen regression avulla saadaan vielä tarkennettua rajausruutuja, jotta ne ovat oikean kokoisia kappaleen ympärillä. [22, 23]

Fast R-CNN. R-CNN:ssä huomattiin muutamia ongelmia. Opettaminen tapahtuu monella tasolla, mikä vie aikaa ja tilaa. Lisäksi kappaleen tunnistus on hidasta. Fast R-CNN pyrkii ratkaisemaan näitä ongelmia. Suurin syy R-CNN:n hitauteen on, että se ei jaa laskentaa. Fast R-CNN:n malli on esitetty Kuvassa 7. Fast R-CNN antaa CNN:lle koko kuvan, kun R-CNN antoi jokaisen rajausruudun erikseen. Tämä on tehty niin, että viimeinen *pooling* kerros on korvattu RoIPoolilla, joka mahdollistaa jaetun laskennan. Lisäksi R-CNN:ssä piirteiden irrottaminen, luokittelu sekä rajausruutujen tarkentaminen tapahtuivat kaikki omissa osissaan. Fast R-CNN yhdistää kaikki kolme osaa, kuten Kuvassa 7 on esitetty. SVM luokittelu on korvattu *softmax* kerroksella. Lisäksi rajausruutujen luominen toimii rinnakkain *softmaxin* kanssa. [23, 24]



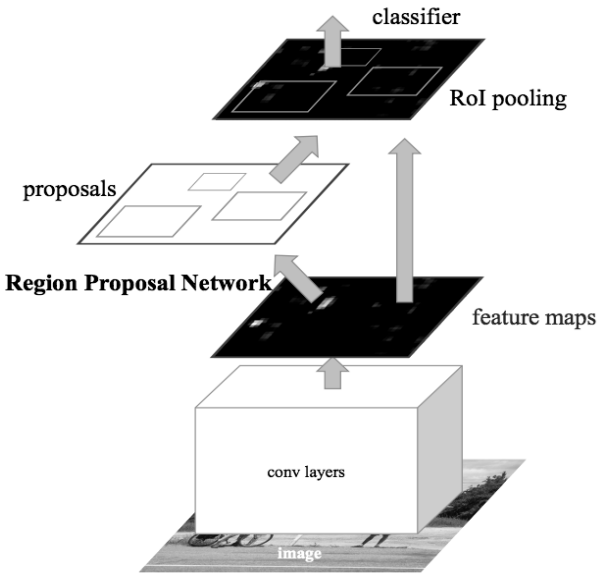
Kuva 7. *Fast R-CNN:n malli [25].*

Faster R-CNN. R-CNN ja Fast R-CNN eivät kiinnitä huomiota alueen ehdotuksiin, joiden tekeminen on myös hidasta. Faster R-CNN parantaa alueen ehdotuksia *region proposal networkilla* (RPN), joka käyttää samaa CNN:ää, jota käytetään piirteiden irrottamiseen. RPN malli on esitetty Kuvassa 8. RPN on lisätty viimeisen konvoluutiokerroksen jälkeen. Se tekee ehdotukset siis piirrekartan perusteella. RPN ajaa liukuikkunan piirrekartan yli. Jokainen ikkuna antaa k-määrän potentiaalisia rajausruutuja ja pisteytykset jokaisen laatikon laadulle. Laatikot ovat eri kokoisia ja niillä on eri kuvasuhteet. Näitä laatikoita kutsutaan ankkurilaatikoiksi *(anchor box).* [23, 26]



Kuva 8. *Region proposal networkin malli [26].*

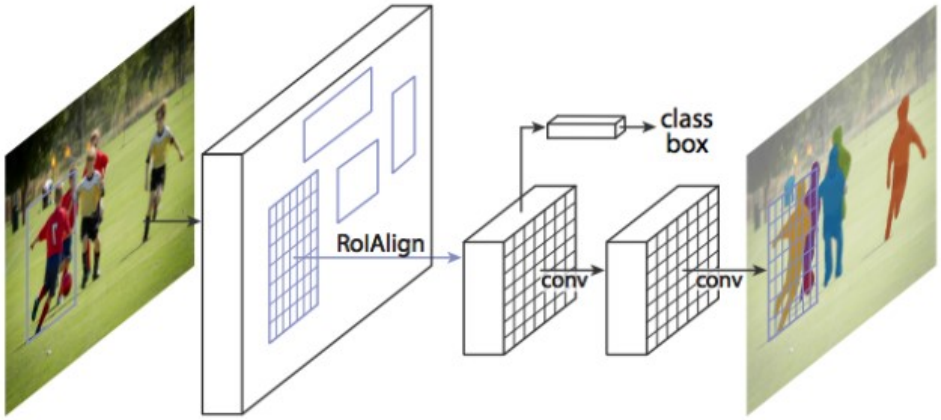
Jokaiselle ankkurille on määritetty, onko siinä kappale vai ei. Tämä tieto menee luokittelu kerrokselle. Ankkurille määritetään kappale kahdella tavalla: ankkurilla on korkein leikkauksen ja yhdisteen suhde todellisen ruudun *(ground truthin)* kanssa tai leikkauksen ja minkä tahansa todellisen ruudun suhde on yli 0,7. Ankkurien koordinaatit menevät rajausruutujen kerrokselle ja todennäköisyydet, onko ankkurissa kappaletta vai ei, me-



Kuva 9. *Faster R-CNN:n malli [26].*

nevät luokittelukerrokselle. Osa ehdotuksista voi mennä päällekkäin. Tätä on vähennetty käyttämällä *non-maximum suppression* menetelmää. [23, 26] Faster R-CNN malli on esitetty Kuvassa 9.

Mask R-CNN. Edelliset R-CNN menetelmät antoivat kappaleille vain rajausruudut, kun Mask R-CNN pyrkii pelkän rajausruudun sijaan antamaan tarkat pikselit, joissa kappale on. Mask R-CNN lisää Faster R-CNN:ään haaran, joka antaa binaarisen maskin. Tämä maski kertoo joka pikselille, onko kyseisessä pikselissä osa jotain kappaletta vai ei. Faster R-CNN RoIPool kerroksia on muokattu Mask R-CNN:ään, koska RoIPoolin antama piirrekartta on asettunut hieman väärin. Tämä johtuu kuvan kutistamisesta, jolloin pikseliarvot pyöristyvät. Mask R-CNN käyttää RoIPool sijasta RoIAlignia, joka käyttää bilineaarista interpolaatiota, jolloin pyöristystä ei tapahdu ja saadaan tarkat pikseliarvot. [23, 27] Mask R-CNN malli on esitetty Kuvassa 10. Kuvasta 10 nähdään, kuinka Mask R-CNN avulla on saatu pelaajien kaikki pikselit tunnistettua pelkän rajausruudun sijasta. Kuvan vasemmassa laidassa näkyy rajausruutu ja oikeassa laidassa saman rajausruudun sisällä pelaajan pikselit on muutettu oranssiksi.



Kuva 10. *Mask R-CNN:n malli [27].*

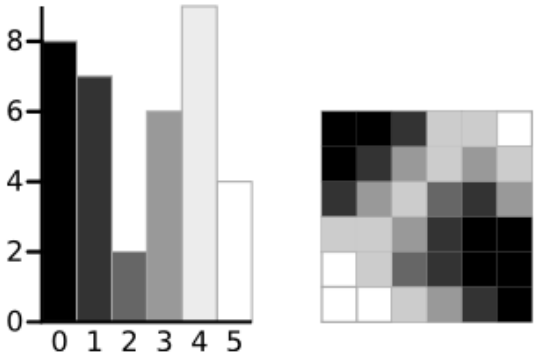
3.2 Kuvan segmentointi

Kuvan segmentointi on menetelmä, jossa kuva jaetaan osiin *(sets of pixels)*. Kuvan segmentoimisen tarkoitus on yksinkertaistaa tai muuttaa kuva helpommin analysoitavaan muotoon. [28] Usein kuvassa on paljon ylimääräistä mukana, jolloin kuvan segmentoimisella saadaan kuvasta erotettua tärkeimmät osat eli esimerkiksi luokiteltava kappale.

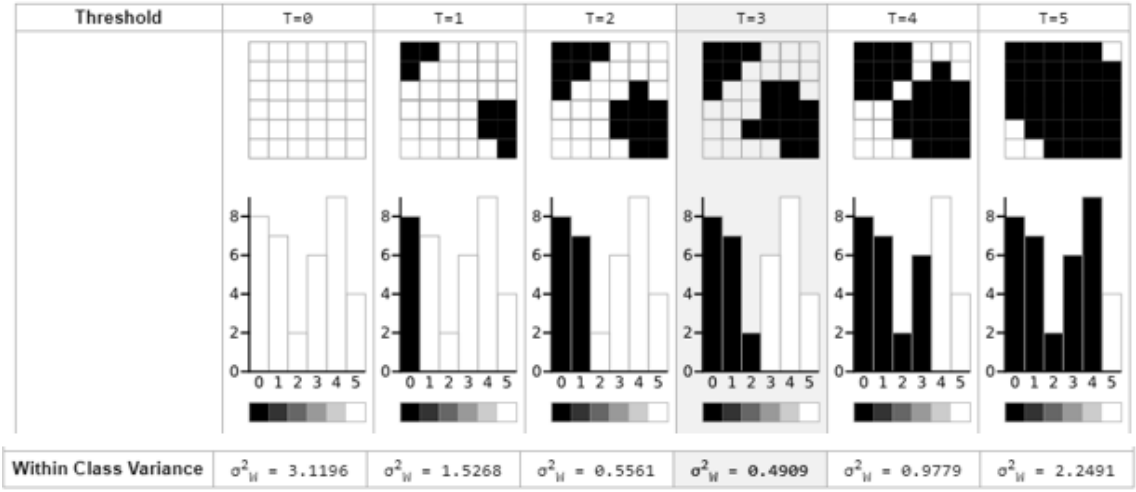
Thresholding. *Thresholding* on yksinkertaisin ja eniten käytetty segmentoimismenetelmä. *Thresholdingia* käytetään yleensä ensimmäisenä vaiheena monissa kuvan analysointimenetelmissä, kappaleen kuvauksessa ja visualisoinnissa. *Thresholdingilla* erotetaan tausta etualasta *(foreground)*. *Thresholding* voidaan määritellä esimerkiksi niin, että kuvan kaikki pikselit, joiden intensiteetti on pienempi kuin jokin vakio, korvataan mustilla pikseleillä. Vastaavasti kaikki pikselit, joiden intensiteetti ylittää vakioarvon, korvataan valkoisilla pikseleillä. [29]

Thresholding menetelmät voidaan jakaa globaaliin ja paikalliseen. Globaalissa käytetään yhtä kynnysarvoa koko kuvaan ja paikallisessa joka pikselille valitaan oma kynnysarvo käyttäen paikallista tietoa. Menetelmät voidaan myös luokitella sen perusteella, mitä tietoa ne manipuloivat. Näitä ovat esimerkiksi histogrammi, ryhmittely *(clustering)*, entropia, kappaleen ominaisuudet, spatiaalinen ja paikallinen. [29]

Otsun menetelmä on yleisimmin käytetty thresholding menetelmä. Se on yksinkertainen, luotettava ja sopeutuva. Otsun menetelmässä iteroidaan kaikkien mahdollisten kynnysarvojen läpi ja lasketaan molempien puolien pikselitasojen leviämiset [28]. Otsun metodi



Kuva 11. *Harmaan tason kuva ja sen histogrammi [28].*



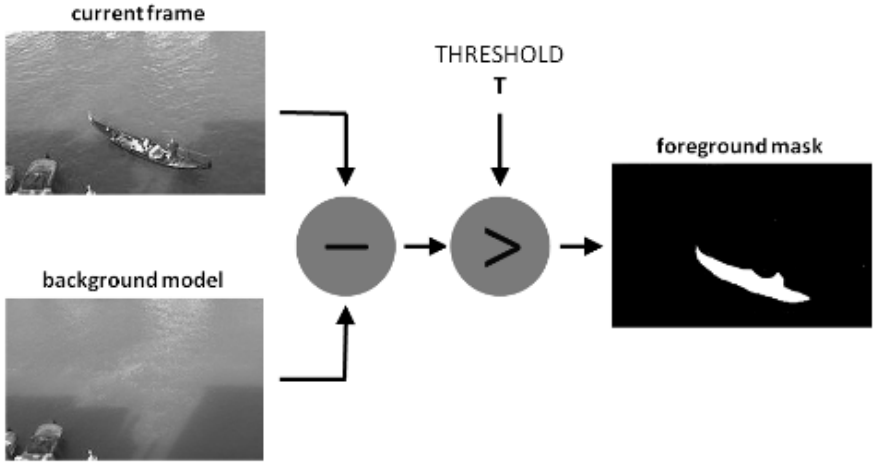
Kuva 12. *Eri kynnysarvojen histogrammit ja varianssit [28].*

valitsee optimaalisen kynnysarvon maksimoimalla harmaiden tasojen luokkien välisen varianssin *(between class variance)* etualassa ja taustassa tai vastaavasti minimoimalla luokkien painotettujen varianssien summan eli sisäisen varianssin *(within class variance)*. [29]

Otsun menetelmässä harmaan tason kuva muutetaan ensin histogrammiksi [28]. Histogrammi ja harmaan tason kuva on esitetty Kuvassa 11. Tämän jälkeen histogrammi jaetaan kahteen osaan ja lasketaan varianssi. Nämä vaiheet tehdään kaikille kynnysarvoille ja näistä valitaan pienin varianssi. [28] Tästä havainnollistava esimerkki näkyy Kuvassa 12. Kuvasta nähdään, että kynnysarvolla 3 saadaan pienin varianssi. Kuvassa 13 on esitetty harmaan tason kuva, joka on muutettu binäärimuotoon Otsun metodilla. Kuvassa on esitetty myös kuvan histogrammi ja laskettu kynnysraja.

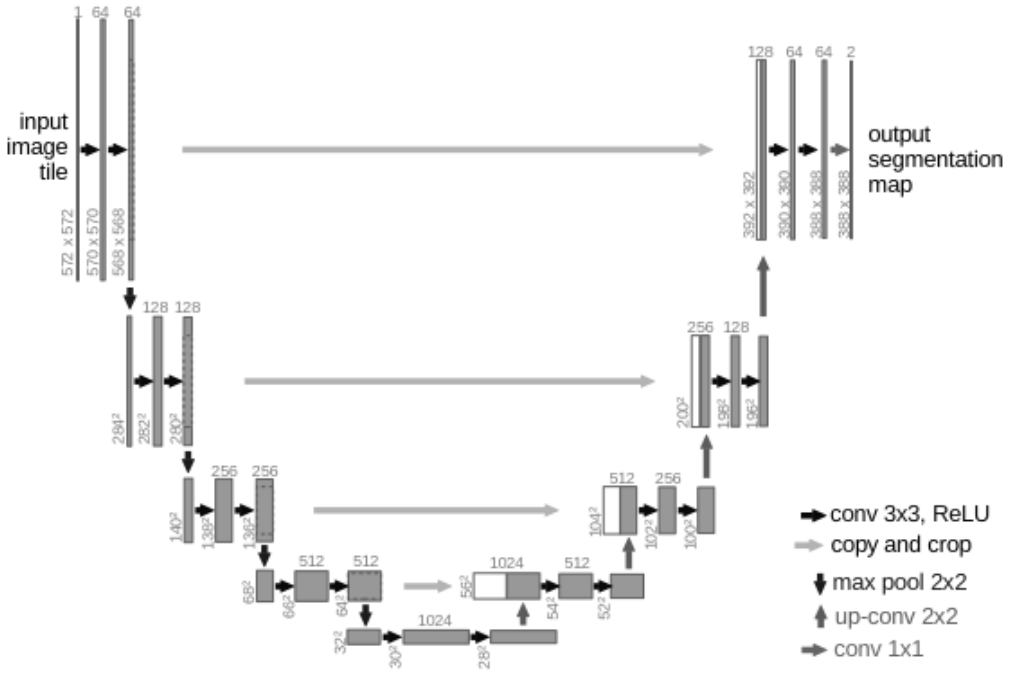
Kuva 13. *Harmaan tason kuva (vasemmalla), binäärinen kuva (keskellä) ja histogrammi (oikealla). Histogrammista nähdään myös kynnysraja [28].*

Background substraction liittyy kappaleen tunnistamiseen videokuvasta. Menetelmä käyttää staattisen taustan matemaattista mallia ja vertaa sitä jokaiseen uuteen kehykseen *(frame)* tai videosekvenssiin. Sen avulla saadaan luotua etualan maski. Etualan maski on binaarinen kuva, joka sisältää ne pikselit, jotka kuuluvat etualaan eli kappaleeseen. Menetelmä siis laskee erotuksen nykyisen kehyksen ja taustan mallin välillä. Havainnollistava kuva *background substractionista* on esitetty Kuvassa 14. *Background substractionissa* on neljä pääaskelta: esikäsittely, taustan mallintaminen, etualan tunnistaminen ja datan validointi. Validointi tarkoittaa vahvistamista tai hyväksyntää. [30-32]



Kuva 14. *Background substractionin idea [32].*

Esikäsittely muuttaa saadun datan muotoon, jota voidaan käyttää mallintamisessa. Mallintamisessa uudet videokehykset päivitetään ja lasketaan taustan malliin, joka toimii staattisena kuvauksena. Etualan tunnistamisessa tunnistetaan kuvasta ne pikselit, joita taustan mallilla ei voida selittää. Tunnistamattomat pikselit toimivat kandidaatteina etualan maskiin seuraavassa vaiheessa. Datan validointi tutkii ja poistaa etualan maskista pikselit, jotka eivät ole haluttuja. [30, 31]



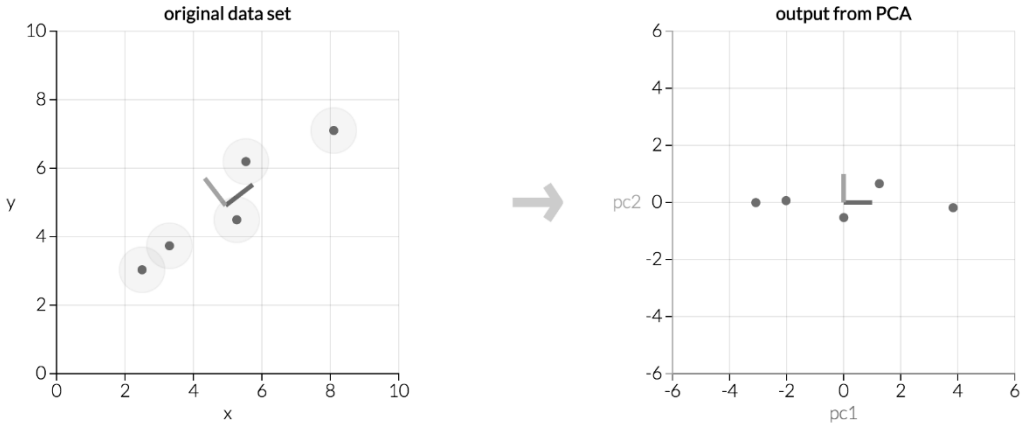
Kuva 15. *U-Netin rakenne [33].*

U-Net. U-Net on konvoluutioneuroverkkomalli, joka on kehitetty lääketieteellisten kuvien segmentoimiseen. Se luottaa vahvaan datan augmentoinnin käyttöön, mutta se voi toimia muutenkin. [33] Augmentointi tarkoittaa, että olemassa olevaa dataa kasvatetaan eri menetelmin, esimerkiksi kääntämällä kuvaa [34]. Saatu data voidaan sen avulla käyttää tehokkaammin. U-Netillä on saatu niin hyviä tuloksia, että sitä käytetään myös muilla aloilla [35].

U-Netin ideana on lisätä tavallisen konvoluutioverkon eli kutistavan *(contracting)* verkon perään kerrokset, missä alinäytteistys-operaatiot *(subsampling)* on korvattu ylinäytteistys-operaatioilla *(upsampling)*. Nämä kerrokset kasvattavat tuloksen resoluutiota. U-Netissä käytetään *skip connections* menetelmää, jossa mallissa on ylimääräisiä yhteyksiä eri kerroksien välillä, jotka ohittavat muita kerroksia [36]. Korkean resoluution piirteet kutistavasta osasta on yhdistetty ylinäytteistys-operaation tuloksen kanssa. Näin konvoluutiokerros voi oppia paremmin paikannetun lopputuloksen. U-Net koostuu siis kutistavasta osasta ja symmetrisestä laajentavasta *(expanding)* osasta. [33] U-Netin malli on esitetty Kuvassa 15. Kuvasta nähdään kuinka rakenne muodostaa U-mallin, josta menetelmä on myös saanut nimensä.

3.3 Kuvan luokittelu

Myös kappaleiden luokittelussa menetelmät jaetaan ohjaamattomiin ja ohjattuihin menetelmiin. Ohjaamattomia menetelmiä ovat esimerkiksi pääkomponenttianalyysi ja *random*



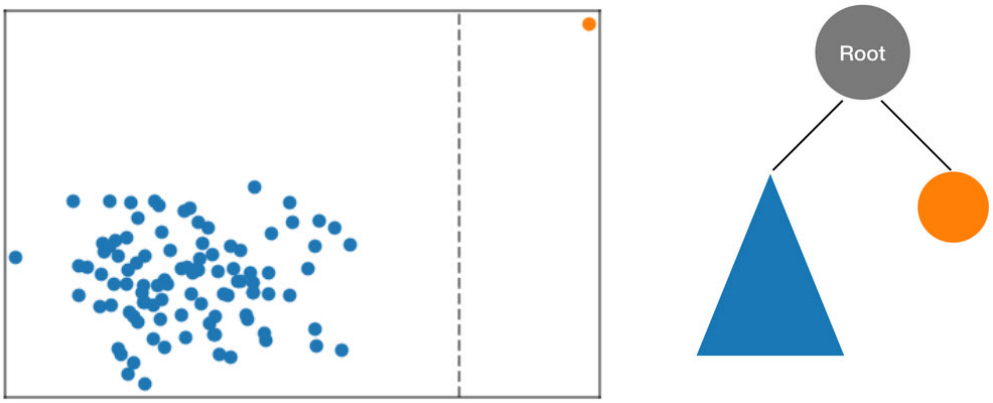
Kuva 16. *PCA:n idea. Vasemmalla on alkuperäinen data ja oikealla PCA:n avulla lasketut komponentit pc1 ja pc2. [37]*

cut forest. Ohjattuja taas ovat *k-nearest neighbour* ja konvoluutioneuroverkot. Konvoluutioneuroverkot käsitellään laajemmin seuraavassa luvussa.

Pääkomponenttianalyysi (PCA) on dimension redusointitekniikka. Se on menetelmä, joka analysoi datataulukon, jossa data on kuvattu muutamien korreloivien riippumattomien muuttujien avulla. Se irrottaa tärkeän tiedon taulukosta ja esittää sen uusina ortogonaalisina muuttujina, joita kutsutaan pääkomponenteiksi. PCA näyttää havaintojen ja muuttujien samankaltaisuuden kuviot pisteinä kartalla. PCA:n tavoitteina on irrottaa tärkeimmät tiedot datataulukosta, tiivistää datakokoelman kokoa säilyttäen vain tärkeimmät tiedot, yksinkertaistaa datakokoelman kuvausta ja analysoida havaintojen sekä muuttujien rakennetta. [38] Havainnollistava kuva PCA:sta ja sen komponenteista on esitetty Kuvassa 16. Kuvasta nähdään, kuinka alkuperäinen data on käännetty ja siirretty origoon. Näin pääkomponentti 1 (pc1) esittää datan mahdollisimman pienellä hajonnalla ja pääkomponentti 2 (pc2) mahdollisimman suurella hajonnalla.

Random cut forest (RCF) tunnistaa datasta poikkeavat arvot. Nämä ovat havaintoja, jotka poikkeavat muuten hyvin säännöllisistä tai kuvioiduista arvoista. Nämä arvot on hyvä saada datasta pois, koska ne monimutkaistavat koneoppimistehtäviä. RCF antaa jokaiselle datapisteelle poikkeamapisteytyksen. Pieni arvo kertoo, että piste on normaali eli sopii joukkoon hyvin. Suuri arvo tarkoittaa, että piste voi olla poikkeava. Suuren ja pienen arvon raja riippuu sovelluksesta. Yleensä ne arvot, jotka ovat keskiarvosta yli kolmen keskihajonnan verran suurempia ovat poikkeavia. [39]

RCF:n pääidea on luoda metsä, jossa jokainen puu saadaan käyttämällä osaa opetusdatasta. Ensimmäinen vaihe RCF:ssä on satunnaisen näytteen ottaminen opetusdatasta. Seuraava vaihe on rakentaa RCF käyttäen satunnaista näytedataa. Ensiksi näytedata jaetaan yhtä moneen samankokoiseen osaan, kuin metsässä on puita. Jokainen

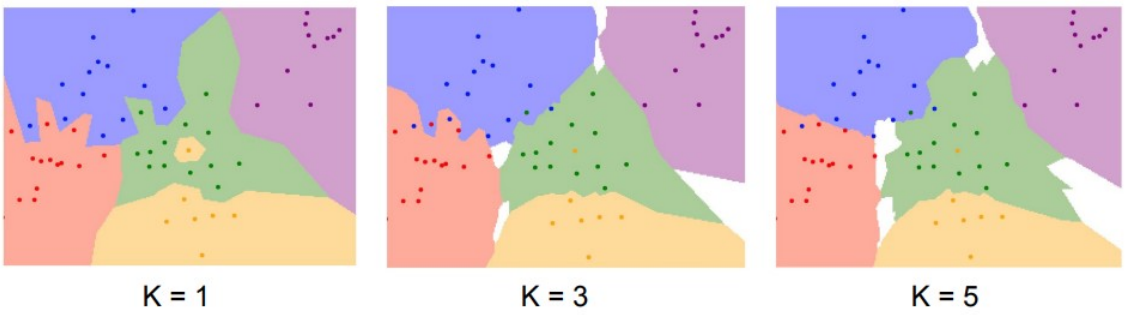


Kuva 17. *RCF rajausruudut ja binääripuu. Poikkeava piste on kuvassa esitetty oranssilla. [39]*

osa muodostaa oman puun. Puu rekursiivisesti organisoi osansa binääripuuksi, osittamalla datan rajausruutuihin. [39, 40] Kuva 17 havainnollistaa asiaa. Kuvassa oranssi datapiste erottuu selkeästi muista datapisteistä.

RCF organisoi Kuvan 17 datan niin, että ensiksi se laskee datan rajausruudun, valitsee satunnaisen dimension (dimensioille, joilla on korkeampi varianssi, annetaan suurempi painoarvo) ja satunnaisesti määrittää hypertason leikkauksen paikan kyseisen dimension läpi. Molemmat leikkauksen luomat osa-alueet määrittävät oman alapuun. Näin poikkeava data todennäköisesti eristäytyy omaan rajausruutuun. Molemmille puun osille lasketaan uudet rajausruudut ja tätä jatketaan, kunnes jokainen puun haara esittää yhden datapisteen näytedatasta. Mitä aikaisemmin piste eristäytyy muista, sitä suuremman poikkeamapisteytyksen se saa kyseiseltä puulta. Lopullinen poikkeamapisteytys on kaikkien puiden antaman pisteytyksen keskiarvo. Mitä vähemmän tasoja tarvitaan eristämään testattava datapiste, sitä todennäköisemmin se on poikkeava. [39, 41]

K-nearest neighbour (KNN) on yksinkertainen ohjatun oppimisen menetelmä, jota voidaan käyttää sekä luokittelu- että regressio-ongelmien ratkaisemiseen. KNN menetelmä olettaa, että samankaltaiset asiat ovat lähellä toisiaan. Se siis laskee datapisteiden etäisyyksiä. KNN käy datan läpi yksitellen ja laskee jokaiselle pisteelle etäisyyden muista pisteistä. Kun kaikki etäisyydet on laskettu, ne pistetään järjestykseen pienimmästä suurimpaan. Menetelmässä on määritetty haluttu K:n arvo. Pienimmistä lasketuista etäisyyksistä valitaan K ensimmäistä arvoa. Jokaisella arvolla on oma tunniste *(label)* ja eniten esiintyvä tunniste on menetelmän veikkaus tunnisteeksi. [42] Kuvasta 18 nähdään kuinka K:n eri arvot vaikuttavat luokkiin. K:n arvolla 1 keltaisen luokan pieni osa on vihreän luokan sisällä, kun taas arvolla 5 kaikki luokat ovat selkeästi omilla alueillaan. K:n arvolla 5 tosin on myös valkoista aluetta, jolle ei ole määriteltyä luokkaa.

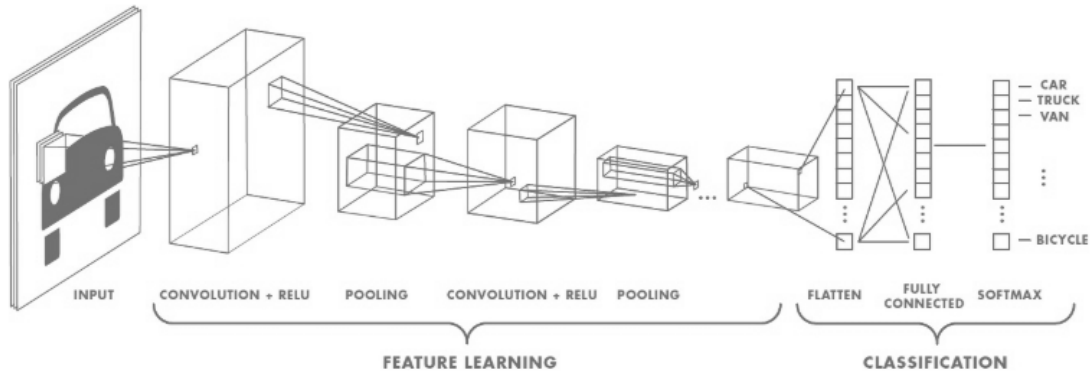


Kuva 18. *KNN eri K arvoilla. Eri värit edustavat eri luokkia. [43]*

3.4 Konvoluutioneuroverkot

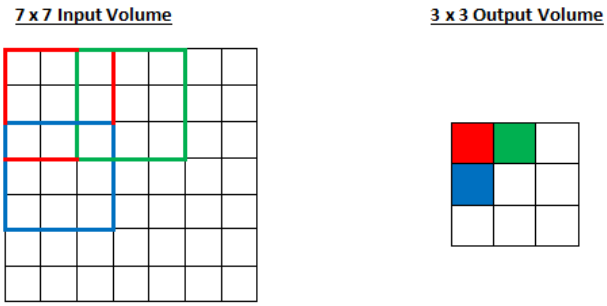
Konvoluutioneuroverkot (CNN) ovat syväoppimisen menetelmiä, jotka ottavat kuvan syötteenä, asettavat tärkeyden eli painot eri muodoille sekä kappaleille kuvassa ja pystyvät erottamaan ne toisistaan. CNN tarvitsee paljon vähemmän esikäsittelyä kuin muut luokittelumenetelmät. CNN:n ideana on supistaa kuvat helpommin käsiteltävään muotoon menettämättä piirteitä, jotka ovat kriittisiä luokittelun kannalta. [3, 44] Konvoluutioneuroverkkoja käytetään kuvan luokittelun lisäksi esimerkiksi puheiden tunnistamisessa, tekstin luokittelussa ja kasvojen tunnistamisessa [45].

CNN koostuu tyypillisesti neljästä operaatiosta, joita ovat konvoluutio, epälineaarisuus, *pooling* ja luokittelu. Näitä operaatioita voi olla useampia peräkkäin. [3, 44] Kuvassa 19 on esitetty esimerkki CNN:n mallista. Kuvasta nähdään, että rakenne on myös jaettu kahteen osaan: piirteiden oppimiseen ja luokitteluun.

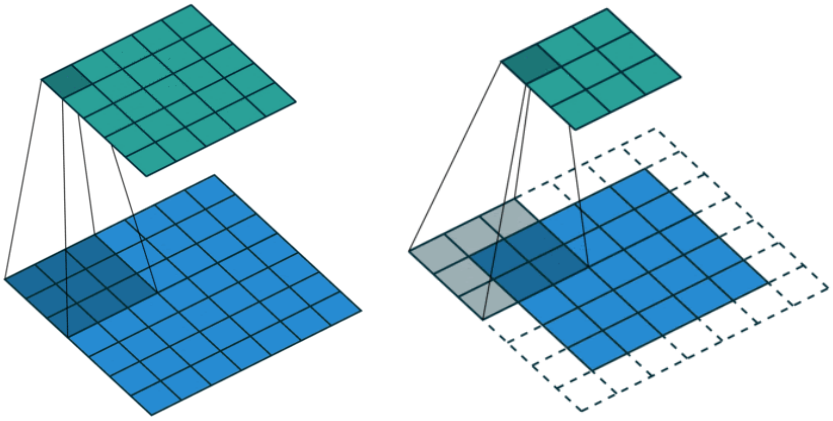


Kuva 19. *Esimerkki CNN mallista [3].*

Ensiksi CNN:n konvoluutiokerros ajaa filtterin kuvan läpi jokaisessa kanavassa (RGB), muodostaen konvoloidut piirteet. Pikselimäärää, jonka filtteri liikkuu, voidaan säätää askelparametrilla *(stride).* [3, 44] Kuvasta 20 nähdään, miten filtteri liikkuu, kun askel parametri on 2. Konvoluutiofiltteri aloittaa syötekuvan yläkulmasta (kuvassa punaisella) ja



Kuva 20. *Filtterin liikkuminen 2 pikselin askeleella [46].*

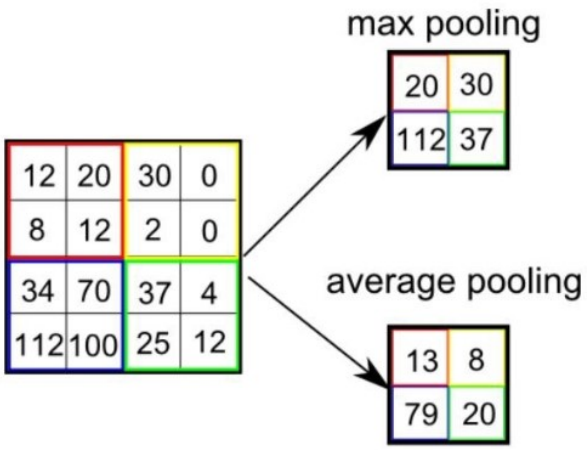


Kuva 21. *Valid padding (vasemmalla) ja same padding (oikealla) [47].*

muodostaa konvoloidun piirteen tuloskuvaan. Askeleella 2 filtteri siirtyy askeleen verran oikealle (kuvassa vihreällä) ja muodostaa tästä konvoloidun piirteen. Näin jatketaan, kunnes koko kuva on käyty läpi. 7x7 syötekuvasta saadaan näin 3x3 konvoloitu tulos.

Konvoluutio-operaatio irrottaa kuvasta korkeatasoisia piirteitä esimerkiksi reunoja. Konvoluutiokerroksia voi olla useampia. Konvoluution tuloksia on kahdenlaisia. Toisessa konvoloidun piirteen dimensio on supistettu ja toisessa sitä on joko kasvatettu tai se on pysynyt samana. Nämä riippuvat täytteestä *(padding)*. *Valid padding* on konvoluutio ilman täytettä, joten se supistaa dimensiota. *Same padding* lisää nollat syötekuvan ympärille, jolloin konvoloidulla piirteellä on sama dimensio kuin alkuperäisellä kuvalla. [3, 44] Kuvassa 21 on esitetty *valid padding* ja *same padding.*

Jokaisen konvoluution jälkeen on aktivointifunktio esimerkiksi *Rectified linear unit* (ReLU). ReLUn tarkoituksena on esitellä epälineaarisuus, koska usein oikea data on epälineaarista ja konvoluutio on lineaarinen operaatio. ReLU korvaa konvoloidun piirteen kaikki negatiiviset arvot nollilla. Muita aktivointifunktioita on esimerkiksi Sigmoid ja hyperpolinen tangentti. [48]

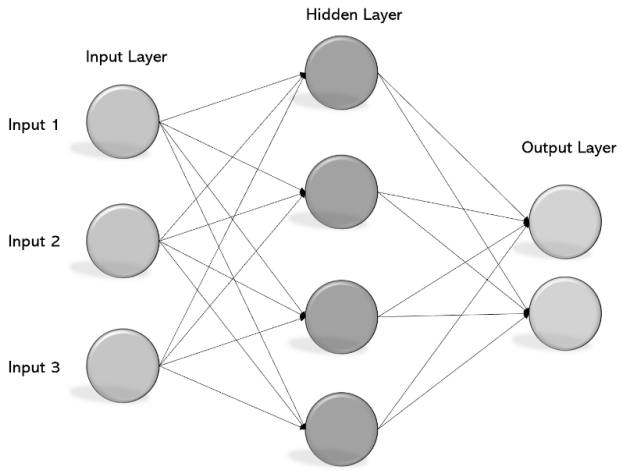


Kuva 22. *Max poolingin ja average poolingin periaate [3].*

Seuraavana on alinäytteistys eli *pooling* kerros, joka supistaa konvoloidun piirteen kokoa. Tämä tehdään, jotta saadaan laskettua tarvittavaa laskennallista tehoa. Se on myös hyödyllinen dominoivien piirteiden irrottamisessa. *Pooling* tyyppejä on erilaisia, joista yleisimpiä ovat *max pooling* ja *average pooling*. *Max pooling* palauttaa maksimiarvon kuvan osasta, jossa filtteri on. *Average pooling* palauttaa vastaavasti keskiarvon. [3, 44] Kuvasta 22 nähdään kuinka *max pooling* ja *average pooling* muodostuvat.

Konvoluutio- ja pooling-operaatioiden lopputulos esittää syötekuvan korkeatasoisia piirteitä. Tulokset litistetään matriisista vektoriksi ja syötetään täysin yhdistetylle *(fully connected)* kerrokselle. Täysin yhdistetyssä kerroksessa edellisen kerroksen kaikki neuronit on yhdistetty kaikkiin seuraavan kerroksen neuroneihin. Täysin yhdistetty kerros on perinteinen monikerroksinen perseptroniverkko *(multilayer perceptron)* eli eteenpäin suunnattu neuroverkko. Sen rakenne on kehitetty ihmisten aivojen neuronien innoittamana. Perseptroniverkko sisältää vähintään kolme kerrosta: syöte kerroksen, piilotetun kerroksen ja ulostulo kerroksen. Piilotettuja kerroksia voi olla useampia. [3, 44] Esimerkki perseptroniverkon rakenteesta on esitetty Kuvassa 23. Verkolle annetaan kolme syötettä ja sillä on kaksi ulostuloa. Kuvassa on yksi piilotettu kerros.

Opetuksen jokaisen iteraation aikana käytetään ohjatun oppimisen *backpropagation* tekniikkaa. Täysin yhdistetty kerros käyttää *softmaxia* aktivointifunktiona. [3, 44] Opetusdata voidaan jakaa osiin eli eriin *(batch)*. Opetuksessa tarvitaan niin monta iteraatiota, että opetusdata on saatu käytyä kokonaan läpi. Epok on ajanjakso, jonka aikana koko opetusdata on käyty kokonaan läpi. Eräkokoa ja epokien määrää voidaan muuttaa. Esimerkiksi, jos opetusdatassa on 1 000 kuvaa ja eräkoko on 100, niin yhden epokin aikana tehdään 10 iteraatiota. [49]



Kuva 23. *Monikerroksisen perseptroniverkon rakenne [50].*

Konvoluutioneuroverkkojen opetusprosessi on seuraavanlainen [51]:

1. Otetaan erä opetusdatasta
2. Annetaan erä neuroverkolle
3. Lasketaan häviö (ero arvioitujen ja oikeiden luokkien välillä)
4. Lasketaan häviöfunktion gradientti
5. Päivitetään painot gradienttien avulla häviön vähentämiseksi
6. Toistetaan 1-5 kunnes kaikki erät on käyty läpi
7. Toistetaan 1-6 kunnes kaikki epokit on käyty läpi

CNN mallien tarkoitus on oppia opetusdata sekä toimia hyvin myös uuden datan kanssa, jota CNN arvioi. Mallin tarvitsee siis oppia opetusdata tarpeeksi hyvin sekä pystyä yleistämään oppimansa uuteen dataan. Yleistämisen ongelmaan liittyy termit *overfitting* ja *underfitting*. *Overfitting* tarkoittaa, että malli oppii hyvin opetusdatan, mutta uuden datan kanssa toimii huonosti. *Underfitting* taas tarkoittaa, että malli toimii molempien kanssa huonosti. [52] *Overfitting* havaitaan esimerkiksi, jos kuvaajaan piirretään opetusdatan ja testidatan tarkkuudet ja testidatan tarkkuus on pienempi kuin opetusdatan tarkkuus. *Overfittingin* estämiseksi voidaan esimerkiksi kasvattaa dataa, käyttää augmentointia, käyttää paremmin yleistäviä malleja, lisätä säännöllisyyttä tai vähentää mallin monimutkaisuutta. [53]

3.5 Datan käsittely

Syväoppimisen mallien toimivuutta voidaan parantaa augmentoimalla dataa. Tämä tarkoittaa, että datakokoelmaan kuviin tehdään pieniä muutoksia, jotta saadaan luotua keinotekoisia kuvia. Varsinkin, jos saatavilla on vain pieni datakokoelma, augmentoinnilla voidaan kasvattaa datakokoelmaa. Yleisiä augmentointimenetelmiä on *flip, rotation, scale, crop* ja *translation*. [34]

Yleensä augmentointi tehdään niin, että opetuksen aikana jokaisella ajanjaksolla eli epokilla, jokaiselle opetetulle kuvalle toteutetaan satunnainen käytössä oleva augmentointi. Näin jokaisella kuvalla voi olla eri augmentointi eri epokeilla.

Flip menetelmällä voidaan kääntää kuvia horisontaalisti tai vertikaalisti. *Rotation* menetelmällä voidaan kääntää kuvia asteittain. *Scale* menetelmällä voidaan skaalata sisään tai ulospäin. *Crop* menetelmällä alkuperäisestä kuvasta otetaan satunnaisesti satunnaisen kokoinen pala ja se muutetaan samaan kokoon kuin alkuperäinen kuva. *Translation* menetelmällä kuvaa liikutetaan x ja y suunnissa. [34]

Muita menetelmiä, joita esimerkiksi Labra.AI:n Visionilla käytettiin, on *blur, multiply* ja kontrastin säätämiset. *Blur* tekee kuvalle Gaussian blur-sumennuksen, jonka avulla kuvasta saadaan poistettua kohinaa ja vähennettyä yksityiskohtia [54]. *Multiply* kertoo kuvamatriisin halutulla vakiolla. Visionissa on kontrastin tasaaja, jolla voidaan tehdä adaptiivinen histogrammin tasaus. Se jakaa histogrammin koko taajuusalueelle. Kontrastia voidaan myös augmentoida eli muuttaa tiettyjen intensiteettien välillä. [55]

3.6 Lajittelusovelluksia

Erilaisia konenäköratkaisuja on käytetty monenlaisissa lajittelusovelluksissa ja linjastoissa jo pitkään, mutta 2010-luvulla konenäön käyttö erilaisissa kohteissa on reilusti yleistynyt tiedon lisääntyessä esimerkiksi konvoluutioneuroverkoista. Konenäköratkaisuja on kokeiltu esimerkiksi useissa erilaisissa hedelmien ja vihannesten sekä muiden maataloustuotteiden lajitteluissa [1, 56]. Muita suosittuja kohteita ovat esimerkiksi erilaisten metalli-, keraami- ja posliinipintojen vikojen tunnistaminen sekä tarkastaminen. [2, 57, 58]

Monissa tapauksissa lajittelua käytetään viallisten tuotteiden hylkäämiseen, laadun erotteluun tai yksinkertaisesti eri tuotteiden erotteluun. Monessa lajittelusovelluksessa ongelmana on ollut ihminen, joka voi tehdä virheitä lajittelussa. Ihmisen lajitteleman tuotteen laatu voi olla vaihtelevaa. Jossain tapauksissa ihmisen ei ole edes turvallista tai mahdollista tehdä lajittelua. Manuaalinen lajittelu ei jossain tapauksissa ole tarpeeksi kattavaa. Esimerkiksi joitain terästuotteiden pintoja tarkastetaan manuaalisesti niin, että

siitä leikataan 30 m pätkä satunnaisesti, jonka ammattilainen tarkastaa. Tämä kattaa kyseisen teräksen pinnasta vain 0,05 %. Lisäksi lajittelutyö voi olla yksitoikkoista ja tylsää ihmiselle, jolloin manuaalisesta työstä halutaan automaattista. Tällöin on haettu usein ratkaisua konenäöstä. [1, 2, 56]

Lajittelusovelluksissa on käytetty paljon erilaisia luokittelumenetelmiä. Jotkut voivat olla hyvinkin monimutkaisia sekä monivaiheisia, mutta varsinkin uusimmissa sovelluksissa ne ovat hyvinkin yksinkertaisia. Yleisimmin luokittelumenetelmänä on kuitenkin käytetty erilaisia neuroverkkoratkaisuja. [1, 2, 56]

Suurimpana erona erilaisten lajittelusovellusten ja katodien lajittelun välillä on olosuhteet. Suurimmassa osassa lajittelusovelluksia on linjastolle tehty optimiolosuhteet, jolloin kameran ottamat kuvat olisivat aina samassa valaistuksessa ja ne olisi otettu samassa paikassa. Todella monessa sovelluksessa on myös käytössä liukuhihna ja liikkuva kappale. Monessa sovelluksessa kappale liikkuu liukuhihnalla ja varsinkin metallien pintavikojen tunnistusratkaisuissa liukuhihnan vauhti on todella nopea, jolloin kuvan käsittelyn pitää tapahtua erittäin nopeasti. Katodien lajittelussa käsitellään staattista kappaletta. Monissa sovelluksissa on käytetty neuroverkkoja, joille on annettu parametrina esimerkiksi kuvasta mitattuja suureita. Konvoluutioneuroverkot ottavat parametrina koko kuvan, jolloin esikäsittelyä tarvitaan vähemmän. [1, 2, 56]

Taulukossa 1 on esitetty muutama erilainen lajittelusovellus, jossa on käytetty konenäköä. Taulukossa on listattuna mitä on lajiteltu, minkä perusteella lajittelua on tehty, moneenko luokkaan on lajiteltu, miksi on lajiteltu sekä mitä menetelmää on käytetty. Taulukossa CNN, ANN, BPNN ja RVM ovat lyhenteitä sanoista: *convolutional neural network, artificial neural network, backpropagation neural network ja relative vector machine.* Taulukosta nähdään, että monesti lajittelun perusteena on kappaleen ulkonäkö. Tämä tarkoittaa, että usein syynä on asiakastyytyväisyys.

Ensimmäisenä taulukossa on mangostanin lajittelu pinnan virheiden perusteella. Mangostanin lajitteluun on hyödynnetty konenäköä samasta syystä kuin monessa muussakin sovelluksessa eli ihmisen takia. Ihmisen tekemä manuaalinen lajittelu on kallista, se hidastaa prosessia ja tuotteen laatu vaihtelee. Mangostanit luokitellaan hyviin ja huonoihin. Tässä sovelluksessa mangostanit on lajiteltu ensin manuaalisesti hyviin ja huonoihin, jonka jälkeen niistä on otettu kuvat. Kuvat on muutettu samankokoisiksi ja sen jälkeen syötetty konvoluutioneuroverkolle, joka opettelee manuaalisesti luokitellut kuvat. Testissä käytettiin 90 hyvää mangostanin kuvaa ja 30 huonoa kuvaa. Tarkkuudeksi testissä saatiin konvoluutioneuroverkon avulla 97 %. [59]

Taulukko 1. *Erilaisia lajittelusovelluksia.*

Kappale	Lajitteluperuste	Luokat	Miksi lajitellaan?	Menetelmä	Vuosi
Mangostani	Pinnan virheet	2	Ulkonäkö	CNN	2017 [59]
Manteli	Muoto	5	Ulkonäkö	ANN	2015 [60]
Metalliromu	Metallityyppi	3	Eri metalleja	Data analyysi, ANN	2010 [61, 62]
Omena	Väri	5	Ulkonäkö	NN	1997 [63]
Peruna	Koko, muoto	5	Ulkonäkö	Data analyysi	2012 [64]
Posliini	Viallisuus	2	Viallisten poisto	Monia menetelmiä, CNN paras	2017 [65]
Posliini	Merkkikoodi	18	Tunnistetaan merkit	CNN	2018 [57]
Siipikarja	Kokonaisuus	2	Kokonaisten erottelu	BPNN	1996 [66]
Teräspalkki	Pinnan virheet	5	Virheiden luokittelu	BPNN, RVM	2010 [67]
Vehnä	Muoto	2	Jauhon laatu	ANN	2016 [68]

Mantelien lajittelusovelluksessa mantelit luokiteltiin viiteen luokkaan niiden muodon perusteella. Konenäön käyttöön on syynä ihmisten tekemän työn korkeat hinnat, hitaus ja tarkkuus. Lisäksi työn tekeminen vaatii kokenutta työntekijää, joka pystyy luokittelemaan mantelit. Mantelit luokitellaan normaaleihin, rikkinäisiin, ryppyisiin, tuplamanteleihin ja mantelin kuoriin. Mantelit on manuaalisesti lajiteltu luokittain ja niistä on otettu kuvat niin, että kuvassa näkyy vain yhden tyyppisiä manteleita kerrallaan. Tämän jälkeen kuvista on segmentoitu mantelit. Segmentoinnin jälkeen kuvista erotetaan hyödyllisiä muodon, värin ja tekstuurin piirteitä, joista PCA:n avulla parhaat annetaan syötteenä neuroverkolle. Mantelit onnistuttiin luokittelemaan 98 % tarkkuudella. [60]

Metalliromun lajittelusovelluksessa erotellaan valettu alumiini, taottu alumiini ja magnesium toisistaan. Sovelluksessa on hyödynnetty neuroverkkoja, jotta on saatu vanhan automaattisen lajittelulinjan data-analyysiin menevää aikaa pienennettyä ilman tarkkuuden menetystä. Järjestelmässä hyödynnetään liukuhihnaa ja vaakaa. Kappaleista mitataan muutamia muotoparametreja, kuten tilavuus, pinta-ala, leveys ja korkeus. Diskriminanttianalyysin ja muotoparametrien avulla kappaleet saadaan luokiteltua. Neuroverkkoja on hyödynnetty korvaamalla viimeisimpiä osia diskriminanttianalyysistä. Metalliromu kappaleita testissä oli noin 1750. Tarkkuus sovelluksessa oli 85 %. [61, 62]

Omenan lajittelussa omenat lajiteltiin viiteen eri luokkaan värin perusteella. Konenäköä hyödynnettiin, koska värierot ovat niin pieniä, että ihmiselle värien erottaminen on vaikeaa. Omenan koko pinnankuva saatiin asettamalla omena kääntyvän pöydän päälle. Yhden omenan kuvaamiseen tarvittiin 20 käännöstä. Kuvista eroteltiin eri väriarvoja, jotka annetiin syötteenä neuroverkolle. Tarkkuudeksi saatiin 95 %. [63]

Perunat luokiteltiin koon ja muodon perusteella ensin hyviin sekä huonoihin ja sen jälkeen hyvät vielä koon perusteella eri kokoihin. Perunan pakkaamisessa konenäköä hyödynnettiin nousevien työvoimakustannusten, manuaalisesta lajittelusta johtuvien tuotantohäviöiden, vaihtelevan laadun, nopeuden ja tarkkuuden takia. Luokittelua varten etsittiin mahdollisimman kattavasti erikokoisia ja -muotoisia perunoita. Luokittelu tapahtui valaistussa teräslaatikossa, jossa liukuhihnan avulla perunat saatiin pyöritettyä niin, että niistä saatiin joka suunnasta kuva. Luokittelussa käytettiin diskriminanttianalyysiä, jolle laskettiin ensin koko- ja muotopiirteitä. Mallille opetettiin 190 normaalia perunaa ja 38 epämuodostunutta perunaa. [64]

Taulukon ensimmäisessä posliinisovelluksessa posliinilautaset lajiteltiin hyviin ja huonoihin. Konenäköä hyödynnettiin, koska manuaalinen lajittelu on kallista, nopeutta rajoittava tekijä, vaatii kokeneen työntekijän sekä altis ihmisen tekemille virheille. Tässä sovelluksessa käytettiin integroitua robotin näköä (iRVision). Sovelluksessa testattiin monta eri mallia, joista konvoluutioneuroverkot osoittautuivat parhaaksi. Kuvat otettiin liukuhihnalla ja robotti poisti hihnalta vialliset lautaset. Mallille opetettiin 200 viallista lautasta ja 146 hyvää lautasta. Konvoluutioneuroverkoilla päästiin 89 % tarkkuuteen. [65]

Toisessa posliinisovelluksessa lautasista tunnistettiin numeroita, kirjaimia ja symboleita, joita oli yhteensä 18. Ensin merkit segmentoitiin ja sen jälkeen tunnistettiin. Luokittelussa käytettiin konvoluutioneuroverkkoja. Kuvia opettamiseen otettiin 17 lautasesta ja saaduista kuvista 75 % käytettiin opettamisessa ja loppuja testaamisessa. Tarkkuudeksi sovelluksessa saatiin 99 %. [57]

Siipikarjasovelluksessa eroteltiin terveitä ja sairaita kanojen ruhoja neuroverkkojen avulla. Konenäköä käytettiin, koska monet tarkastamis- ja lajittelutehtävät ovat toistuvaa ja tylsää työtä. Työn tehokkuus riippuu ihmisen tehokkuudesta. Neuroverkoille syötettiin spektraalisia kuvia. Sovelluksessa käytettiin 21 tervettä ja 70 sairasta lintua. Kun neuroverkolle syötettiin spektraalisia kuvia, tarkkuus oli 93,3 %. Fast Fourierin (FFT) kuvien intensiteettejä syötettäessä tarkkuudeksi saatiin 90 %. [66]

Teräspalkki-sovelluksessa tarkasteltiin pinnan virheitä. Sovelluksessa käytettiin konenäköä, koska teräspalkki liikkuu niin nopeasti, että ihmiselle tehtävä olisi mahdoton. Lisäksi kuvat ovat vääristyneitä ja niitä joudutaan esikäsittelemään ennen luokittelua.

Luokittelu on jaettu neljään osaan. Ensin kuva luokiteltiin sen mukaan, onko kuvassa pinnan virheitä vai ei. Seuraavaksi pinnan virheet luokiteltiin kahteen osaan. Toisesta osasta luokiteltiin saumavirheet sekä rullausjäljet ja toisesta osasta naarmut ja ylitäytöt. Luokitteluissa testattiin neuroverkkoja ja tarkkuudeksi saatiin 91 %. [67]

Vehnäluokittelussa vehnät lajiteltiin leipä- ja durumvehniin. Molemmista luokista sovelluksessa käytettiin 100 kuvaa. Kuvista irrotettiin 21 erilaista visuaalista piirrettä, joista 12 pääpiirrettä annettiin neuroverkolle. Tarkkuudeksi luokittelussa saatiin parhaimmillaan yli 99 %. [68]

4. MENETELMÄT

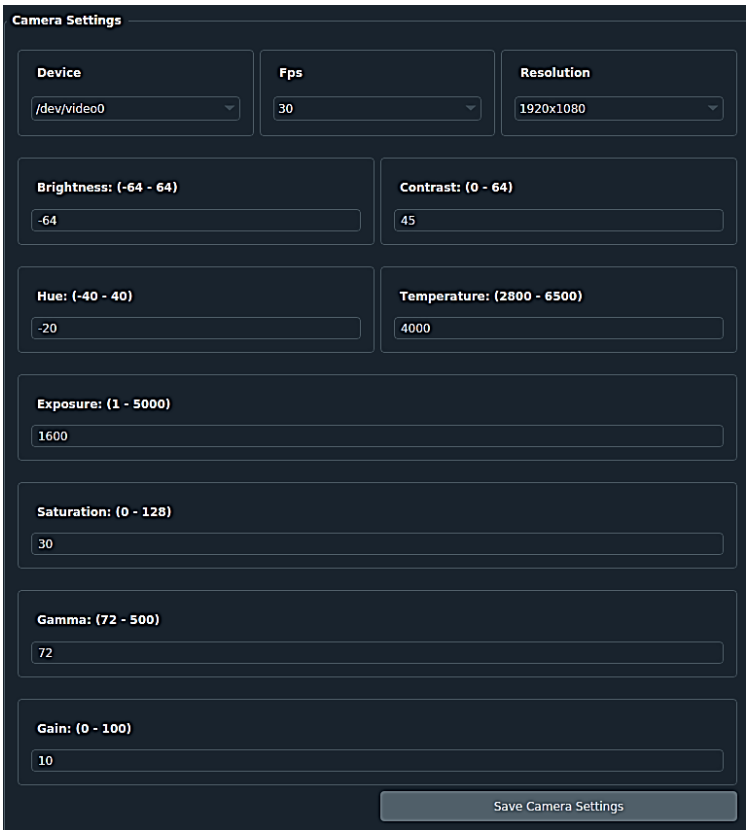
Menetelmät luvussa käydään ensin läpi, että miten data on kerätty. Sen jälkeen käydään läpi Vision-ohjelman opetus- ja testausvaiheista. Näiden jälkeen on käyty läpi Tensorflow-kirjaston avulla tehdyt luokittelut ja testit. Lopuksi luvussa on käyty läpi 4-tuuman palan lajittelutestiä.

4.1 Datan kerääminen

Data kerättiin Labra.AI:n toimittamalla laitteistolla ja Vision-ohjelmistolla. Sen avulla saatiin kuvista tunnistettua ja segmentoitua pelkästään nikkelikatodit. Samaa dataa käytettiin myös tätä diplomityötä varten tehdyssä ohjelmoinnissa.

Kamera haluttiin asentaa suoraan katodinipun päälle ja imukuppinostimen runkoineen täytyi mahtua sen alle, joten kameralle rakennettiin teline. Lisäksi nippujen paikkoja siirrettiin niin, että vain yksi nippu eli lajiteltava nippu näkyi kameran kuvassa. Jotta testijärjestelyt eivät estäneet normaaleja tuotantotoimia, tietokone sijoitettiin mahdollisimman sivuun ja tarvittavat johdot vedettiin yläkautta. Näin myös trukki pääsi operoimaan johtojen alapuolella, eikä voinut ajaa niiden päältä. Johdot oli lisäksi kieritetty huomionauhalla. Keltamusta teline sekä johtojen ylitys on esitetty Kuvassa 24.

Kuva 24. *Kameralle asennettu teline vasemmalla sekä telineestä viety johtojen ylitys oikealla.*



Kuva 25. *Kameran säädettävät parametrit.*

Ennen datan keräämistä, kamera ja ohjelmisto kalibroitiin. Kameran parametrit, joiden avulla kuvaa säädettiin, näkyvät Kuvassa 25. Parametrit ovat kirkkaus, kontrasti, värisävy, lämpötila, valotusaika, kyllästys, gamma ja vahvistus *(gain)*. Aluksi parametreja oli vain viisi, mutta kuvan säätäminen niiden avulla oli haastavaa. Tästä syystä kyllästys, gamma ja kasvatus lisättiin Labra.AI:n toimesta, jotta kuvaa saatiin paremmin säädettyä.

Kirkkausparametri säätää kuvan kirkkautta kasvattamalla kuvan pikseliarvoja tietyllä vakiolla. Kontrastissa kuvan pikseliarvojen erotus kuvan keskiarvoiseen pikseliarvoon jaetaan keskiarvoisella pikseliarvolla, jonka jälkeen se kerrotaan vakiolla. Tämä saa tummat alueet näyttämään tummemmilta ja vaaleammat vaaleammilta. Kontrastin avulla saadaan kuvasta erottumaan pienetkin sävyerot.

Värisävyn muuttaminen tarkoittaa, että pikselien taajuutta pienennettäessä kuva siirtyy lähemmäs sinistä sävyä ja kasvattaessa lähemmäs punaista. Lämpötila on samankaltainen parametri kuin värisävy, mutta se säätää pikselien sinistä ja punaista kanavaa. Kun lämpötilaa kasvatetaan sinisen kanavan arvot pienenevät ja punaisen kanavan arvot kasvavat. Lämpötilaa pienentäessä sinisen arvot kasvavat ja punaisen pienentyvät.

Valotusaika kertoo, kuinka kauan jokaista kameran kehystä valotetaan. Lyhyellä valotusajalla kuvasta tulee tummempi. Liikkeessä oleva kappale näkyy tarkempana lyhyellä valotusajalla, koska kappale ehtii liikkua lyhyemmän matkan valotuksen aikana. Pitkällä

valotusajalla kuvasta tulee kirkkaampi, mutta liikkeellä oleva kappale ehtii liikkua pidemmän matkan, joka aiheuttaa kuvan sumentumista.

Kyllästysparametri säätää kuvan harmautta. Kyllästystä kasvattaessa kuvan pikseleistä poistetaan harmaita sävyjä ja pienennettäessä niitä kasvatetaan. Tämä tarkoittaa siis, että kuvan RGB-kanavien arvoja muutetaan toisiinsa nähden. Kun kyllästys on minimissä, kuva muuttuu mustavalkoiseksi, koska kaikkien kanavien arvot pikseleissä ovat samat. Vastaavasti maksimissa kanavien arvot ovat mahdollisimman kaukana toisistaan.

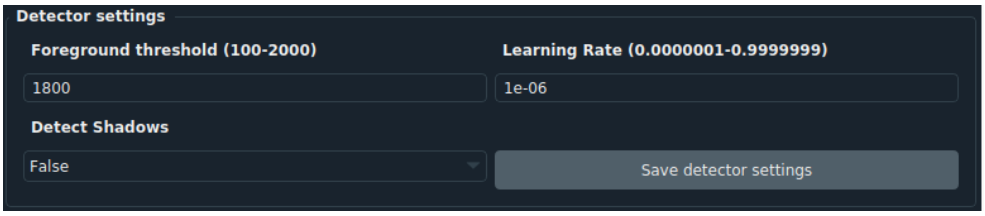
Gamma laskee jokaisen pikselin kaavan $pikseli = pikseli^{(1/gamma)}$ mukaan. Tämä saa tummat ja vaaleat alueet erottumaan kuvasta paremmin, aivan kuten kontrastikin. Gamman säätäminen aiheuttaa ylivalottumista, jolloin kuvan vaaleat alueet saavat maksimiarvonsa ja sävyerot katoavat. Vahvistus taas kasvattaa kuvasensorin jännitettä. Näin jokaisen pikselin signaali vahvistuu, mikä näkyy kuvan kirkastumisena. Jännitteen kasvattaminen lisää kuvaussensorin häiriötä, mikä näkyy kuvassa kohinana.

Aina kun kameran parametreja ja muita arvoja muutetaan, on tausta nollattava. Tämä tarkoittaa, että katodit ja tässä tapauksessa myös trukkilavat poistetaan kameran alta, jotta nollaaminen voidaan tehdä. Tämä hidasti työskentelyä, koska välillä parametreja saatettiin muuttaa kesken katodinipun, jolloin nippu piti ajaa trukilla pois nollauksen ajaksi ja takaisin sen jälkeen.

Tunnistimen *(detector)* asetuksissa säädettävissä on etualan kynnysarvo *(foreground threshold)*, oppimisaste *(learning rate)* ja varjojen tunnistus *(detect shadows)*. Etualan kynnysarvo on raja, kuinka paljon eroa on kappaleen ja taustan värillä. Oppimisaste määrittää kuinka paljon tunnistin oppii jokaisesta uudesta kehyksestä taustan malliin, kun sen edessä ei ole yhtään tunnistettua staattista kappaletta. Tunnistimen asetukset on esitetty Kuvassa 26.

Kun tausta on opetettu ja kuvattava kappale näkyy kuvassa paikallaan, ohjelma tunnistaa sen ja kuvan kappale muuttuu kirkkaammaksi ja tausta vastaavasti tummemmaksi. Ohjelma ottaa tästä kuvan, joka segmentoituu niin, että siinä näkyy vain kyseinen kappale. Kuvassa saattaa myös näkyä jotain pienempiä kirkkaita eli staattisia kappaleita, jotka eivät kuitenkaan ole haluttuja. Tähän apuna on kappaleen koon määritys asetuksissa, jossa voidaan asettaa halutun kappaleen leveyden ja korkeuden minimit sekä maksimit. Nämä arvot ovat pikseliarvoja väliltä 10-1920.

Kappaleen asento ei vaikuta tunnistamiseen. Vision laskee pienimmän mahdollisen suorakulmion, johon segmentoitu kappale mahtuu ja kääntää kuvan siten, että suorakulmion leveämpi sivu asetetaan vaaka-akselille. Vaikka kappale olisi vähän kulmittain kameraan



Kuva 26. *Tunnistimen asetukset.*

nähden, kuva kääntyy samaan asentoon kuin muutkin kuvat. Ohjelma ottaa kuvan aina, kun se tunnistaa uuden staattisen kappaleen, joka on oikean kokoinen. Haluttu koko määritetään ohjelman asetuksissa.

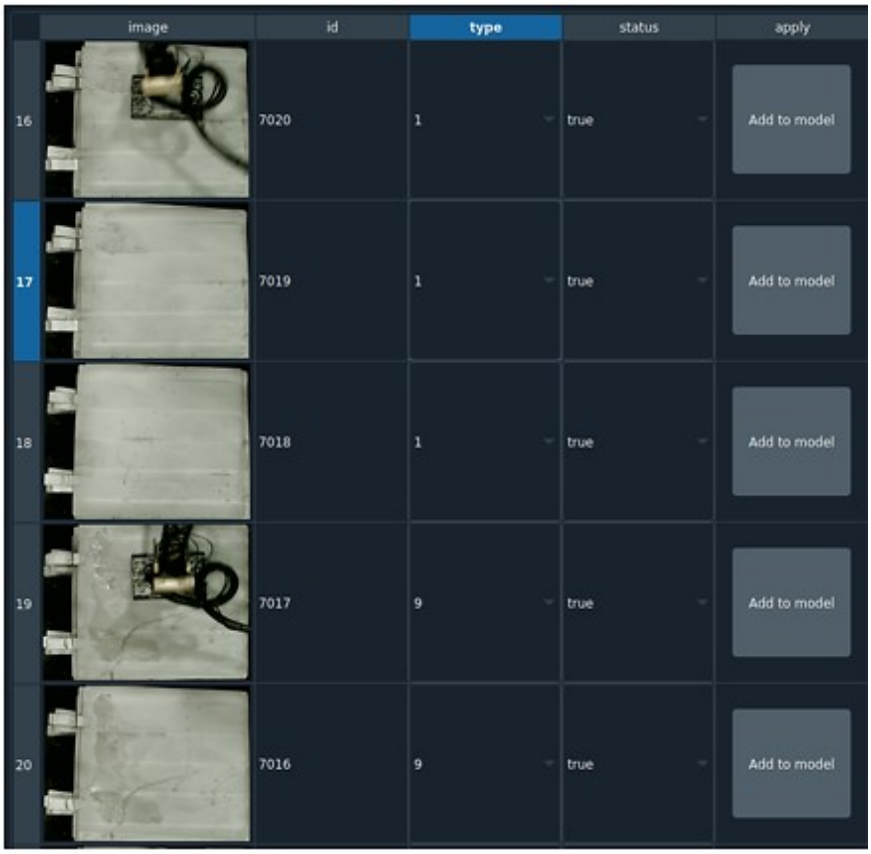
4.2 Visionin opetusvaihe

Visionin luokittelu tapahtui paikan päällä NNH:lla, jossa lajittelua tehtiin samalla manuaalisesti. Visionilla luokiteltaessa, lajittelu tehtiin manuaalisesti kameran ollessa asennettuna manuaalisen lajittelupisteen päälle. Kamera otti kuvan aina nipun päällimmäisestä katodista. Koko nippu käytiin läpi ja jokainen katodi kuvattiin. Tämän jälkeen jokaiselle kuvalle annettiin oikea luokka Visionilla. Luokat olivat priima, standardi ja romu. Luokka valittiin Visionissa valitsemalla *type*-valikosta luokka ja painamalla *Add to model*-nappulaa. Kuva näkymästä on esitetty Kuvassa 27. Kuvassa priima on *type*-valikossa 9 ja standardi on 1. Kuvassa luokkia ei ole vielä opetettu oikein.

Kun mallille oli annettu haluttu määrä kuvia, kuvat opetettiin mallille. Tämän jälkeen mallia päästiin testaamaan. Opetus tehtiin Visionilla painamalla *train model*-nappulaa. Opetuksessa data jaettiin opetusdataan ja testi- eli validointidataan. Validointidata on 20 prosenttia kaikista mallille opetetuista kuvista. Näitä kuvia ei käytetä mallin opetukseen, vaan niillä testataan mallia jokaisen epokin jälkeen.

Opetuksen jälkeen Vision ilmoitti opetuksen tuloksia. Tulokset ovat opetuksen ja validoinnin tarkkuudet sekä häviöt. Tarkkuus kertoo, kuinka monta kuvaa opetukseen käytetyistä kuvista malli pystyy luokittelemaan oikeaan luokkaan suhteutettuna kokonaismäärään. Häviö funktiona on *categorical cross-entropy*, jossa lasketaan kaikkien opetuskuvien todellisen luokan ja ennustetun luokan logaritmisen tulon kokonaiserotus. Opetusvaiheessa malli valitsee sen epokin, jolla validointihäviö on kaikkein pienin.

Aluksi Visionille opetettiin vain kahta luokkaa, jotta varmistettiin ohjelman toimivuus ja parametrit saatiin säädettyä sekä harjoiteltiin ohjelman käyttöä. Luokat olivat priima ja muut. Kun kahdella luokalla alettiin saamaan hyviä tuloksia, siirryttiin kolmeen luokkaan.



Kuva 27. *Laatujen lisääminen malliin.*

4.3 Visionin testausvaihe

Mallia testattiin satunnaisin väliajoin, jonka jälkeen opetettiin taas lisää. Aina mallin opettamisen jälkeen mallia testattiin. Näin saatiin selville, miten malli muuttui edellisestä testauksesta. Visionilla tehdyt testit olivat eri pituisia kuvattujen katodien määrän mukaan. Välillä testattiin vain noin 20 katodia, jolloin pystyi jo toteamaan, että malli tarvitsee opettaa uudelleen. Toisinaan testattiin yli 100 katodia ennen uutta opetusta.

Testauksessa malli laitettiin päälle Visionista ja alettiin lajitella manuaalisesti eri luokkiin samalla tavalla kuin opetusvaiheessa. Vision otti jokaisesta katodista kuvan samalla tavalla kuin opetusvaiheessa, mutta nyt se myös kertoi oman arvionsa katodin luokasta. Aina kun Vision arvioi väärin, mallille opetettiin oikea luokka samalla tavalla kuin opetusvaiheessa. Myös kaikki oikeat arviot olisi voitu lisätä opetusdataan, jolloin datan määrää olisi saatu kasvatettua entisestään, mutta tätä ei tehty ajan säästämiseksi. Kun malli muuttui paljon, malli opetettiin ohjelmalle uudelleen. Mallin ajo on esitetty Kuvassa 28. Kuvan yläreunassa näkyy neljä edellistä kuvaa ja niiden arvioidut luokat.

Kuva 28. *Mallin ajonäkymä. Ylhäällä neljäntenä näkyy, kuinka malli luokittelee kuvan. Kuvasta nähdään myös, kuinka malli ottaa turhiakin kuvia imukuppinostimen ollessa vielä kuvassa.*

4.4 Tensorflowlla tehdyt testit

Tässä työssä muodostettiin omia CNN malleja, jotka toteutettiin Tensorflown avulla. CNN avulla tehtävissä luokitteluissa käytettiin Visionilla kerättyjä kuvia. Tensorflow on avoimen lähdekoodin koneoppimisalusta, jota tässä työssä käytettiin Pythonin Keras ohjelmointirajapinnan avulla [69]. Keras on syväoppimisen ohjelmointirajapinta, joka käyttää Tensorflowta [70]. Se on suunniteltu keskittyen nopeaan käyttöön ja sen tarkoituksena on, että toteutettava idea saadaan tuloksiksi mahdollisimman nopeasti [70]. Kaikki tässä luvussa esitetyt koodit ovat saatavilla osoitteesta: *https://github.com/sodeda/dippa*.

Vision-ohjelmistolla saadut kuvat on jaettu luokittain omiin kansioihin. Visionilla otetut kuvat käytiin läpi ja kuvista poistettiin epäselviä tapauksia, rajatapauksia sekä huonoja kuvia, jotta saatiin selkeytettyä luokkien välisiä eroja. Yhteensä kuvia oli lopulta 640, joista priimalaatua oli 270, standardilaatua 262 ja romua 108. Koska kuvia oli suhteellisen vähän, niiden määrää kokeiltiin kasvattaa keinotekoisesti augmentoimalla. Jokaiselle kuvalle tehtiin yksi tai useampi augmentointi satunnaisesti *vertical flipin, horizontal flipin* ja kirkkauden säädön joukosta. Myös tämä augmentoitu kuva lisättiin dataan, jolloin datan määrä saatiin tuplattua.

Testeissä ensimmäiseksi data luetaan kansioista. Kuvat luetaan OpenCV:n avulla, joka on avoimen lähdekoodin tietokonenäön ja koneoppimisen kirjasto [71]. Kun kuva luetaan, muodostetaan kuvalista, jossa ensimmäisenä alkiona on luettu kuva ja toisena alkiona on tämän kuvan luokka. Kuvan luokka saadaan kansion nimestä. Kuvalista, joka sisältää luetun kuvan ja sen luokan, lisätään datalistaan. Tässä vaiheessa kuvalle tehdään satunnainen augmentointi, joka myös lisätään datalistaan. Datalista sisältää kaikki luetut ja augmentoidut kuvat ja niiden luokat. Funktio, jolla data luetaan ja käsitellään, on esitetty Kuvassa 29.

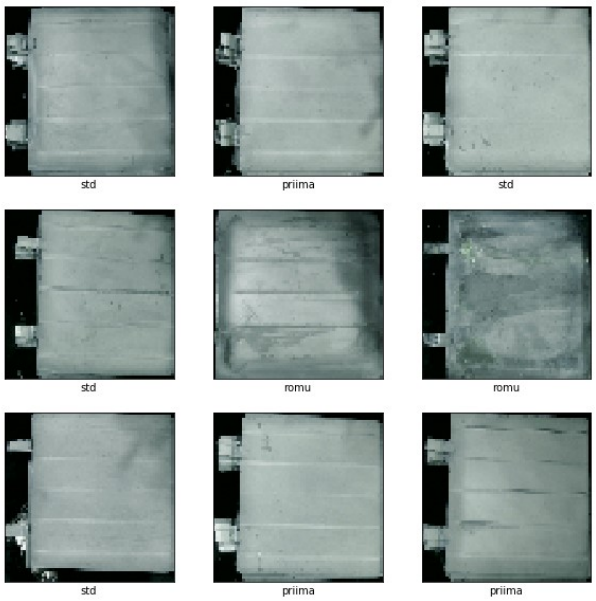
```
def get_data(data, path, label, size):
    for image in os.listdir(path):
        img = cv2.imread(path + '/' + image, 1)
        img = cv2.resize(img, (size, size))
        img_and_label = []
        img_and_label.append(img)
        img_and_label.append(label)
        data.append(img_and_label)
    return data
```

Kuva 29. *Get_data funktio, joka lukee kuvat ja muodostaa datan.*

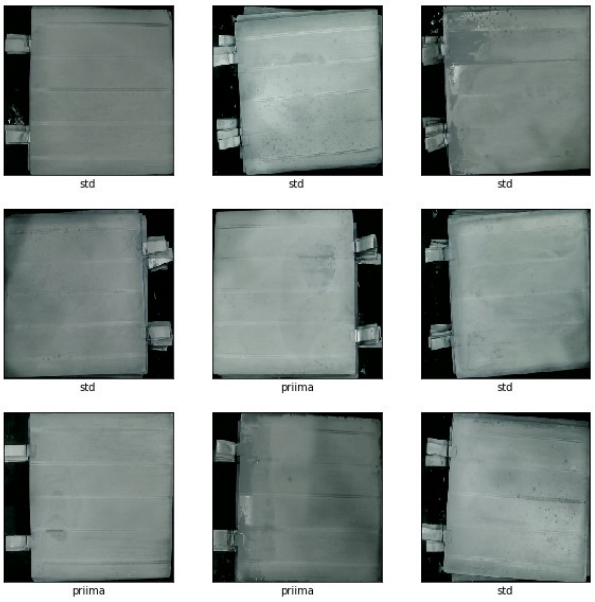
Funktio *get_data* ottaa parametreina datalistan, polun, josta kuvat luetaan, kuvien luokan ja halutun kuvien koon. Ensimmäisellä kerralla *data* on tyhjä lista. Parametri *label* eli luokka, jonka funktio saa on annetussa polussa jokaiselle kuvalle sama. Funktio *get_data* lukee kuvat ja tekee kuvalistan, jossa on kuva ja sen luokka. Jokaiselle kuvalle tehdystä augmentoidusta kuvasta tehdään samanlainen kuvalista, jossa on kuva ja luokka. Kuvalistat lisätään datalistaan, jolloin lopulta saadaan lista kaikista kuvista ja niiden luokista.

Datan hakemisen jälkeen, data on jaettu opetusdataan ja testidataan. Opetus- ja testidatat sekoitetaan, niin että kuvien luokat eivät sekoitu. Tämän jälkeen muodostetaan omat listat kuville sekä luokille erottamalla ne toisistaan. Kuva ensimmäisestä yhdeksästä kuvasta ja niiden luokista sekoitetussa datassa on esitetty Kuvassa 30. Kuvasta nähdään, että luokat eivät ole enää järjestyksessä.

Tämän jälkeen osaa kuvista augmentoidaan vielä uudestaan ennen opetusta. Keras rajapinnassa augmentointi voidaan tehdä *ImageDataGenerator* luokalla. Se luo satunnaisesti käännettyjä kuvia, kun *vertical_flipille* ja *horizontal_flipille* on annettu arvo *True*. Lisäksi kokeiltiin esimerkiksi kirkkauden ja zoomin muuttamista sekä kuvan siirtoa eri suunnissa. Kuvassa 31 on esitetty augmentoinnin jälkeisiä kuvia. Kuvasta nähdään, että esimerkiksi ensimmäisen rivin ensimmäinen ja viimeinen kuva on käännetty *horizontal_flipillä.*



Kuva 30. *Yhdeksän ensimmäistä kuvaa ja niiden luokat.*



Kuva 31. *Augmentoinnin tuloksia.*

Tämän jälkeen valitaan konvoluutioverkon malli. Testeissä on käytetty LeNetiä sekä kustomoituja verkkoja. Kuvasta 32 nähdään miten LeNet malli on luotu. Ensin luodaan *Sequential*-malli, johon lisätään halutut kerrokset halutuilla parametreilla. Konvoluutiokerroksen parametreina on ensimmäisessä kerroksessa käytetty 6 filtteriä, joiden koko on 5, askel koko on 1, aktivointi funktiona on *'tanh'* ja mallille annettava kuvakoko on (32, 32, 3). Eli leveys ja korkeus ovat 32 pikseliä ja värikanavia on 3.

```
def LeNet():
    model = models.Sequential()
    model.add(layers.Conv2D(6, (5, 5), strides=(1,1),
              activation = 'tanh', input_shape = (32, 32, 3)))
    model.add(layers.AveragePooling2D((2,2), strides=(2,2)))
    model.add(layers.Conv2D(16, (5, 5), strides=(1,1),
              activation = 'tanh'))
    model.add(layers.AveragePooling2D((2,2), strides=(2,2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(120, activation = 'tanh'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(84, activation = 'tanh'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(3, activation = 'softmax'))
    return model
```

Kuva 32. *LeNet luokka.*

LeNet-5 on neuroverkkomalli, joka on suunniteltu käsin kirjoitettujen numeroiden tunnistamiseen [72]. Sen rakenne on suoraviivainen sekä yksinkertainen ja tästä syystä se on usein ensimmäinen askel konvoluutioneuroverkon opettamisessa [73]. Sen rakenne on esitetty Taulukossa 2. Taulukosta nähdään, että LeNetissä on kolme konvoluutiokerrosta, joiden välissä on *average pooling*-kerrokset. Näiden jälkeen on täysin yhdistetty kerros ja lopulta luokittelukerros. Aktivointifunktiona toimii *'tanh'*. Mallille annettava kuvan koko on (32, 32).

Taulukko 2. *LeNet-5 rakenne. [73]*

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Kustomoidussa mallissa parametrit, joita säädettiin ja kokeiltiin, olivat syötekuvan koko, eräkoko, eri augmentoinnit, datan laajentaminen, *pooling*, filtterien koko, kerroksien määrä ja tyyppi sekä aktivointifunktio. Mallin pohja on esitetty Kuvassa 33. Kuvassa näkyvässä mallissa on käytetty *max poolingia*. Kerroksien väleihin on laitettu *dropout* kerroksia *overfittingin* estämiseksi.

```
def CustomNet():
    model = models.Sequential()
    model.add(layers.Conv2D(16, (3, 3), activation = 'tanh',
              input_shape = (32, 32, 3)))
    model.add(layers.Conv2D(16, (3, 3), activation = 'tanh'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(16, (3, 3), activation = 'tanh'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(16, (3, 3), activation = 'tanh'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.1))
    model.add(layers.Flatten())
    model.add(layers.Dense(100, activation = 'tanh'))
    model.add(layers.Dropout(0.25))
    model.add(layers.Dense(3, activation = 'softmax'))
    return model
```

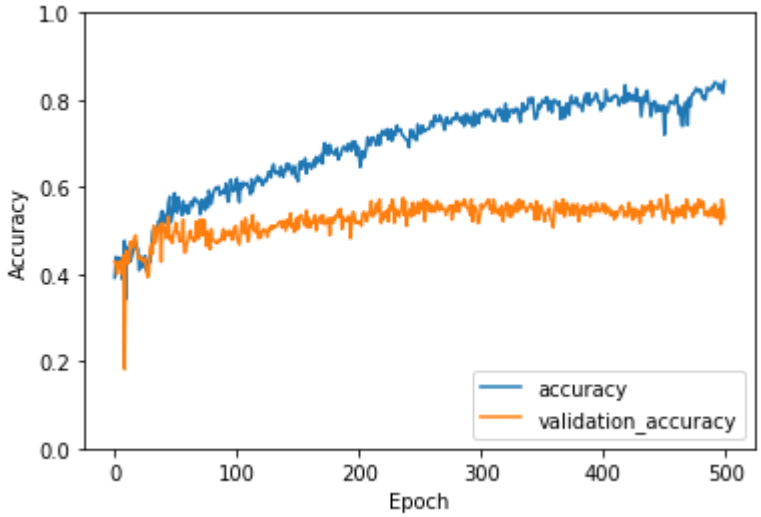
Kuva 33. *Kustomoitu konvoluutioneuroverkkomalli.*

```
5/5 [==============================] - 0s 73ms/step - loss: 0.9061 - acc: 0.6080 - val_loss: 0.9687 - val_acc: 0.5214
Epoch 192/200
5/5 [==============================] - 0s 70ms/step - loss: 0.9144 - acc: 0.6020 - val_loss: 0.9711 - val_acc: 0.5214
Epoch 193/200
5/5 [==============================] - 0s 75ms/step - loss: 0.9231 - acc: 0.5920 - val_loss: 0.9778 - val_acc: 0.5286
Epoch 194/200
5/5 [==============================] - 0s 74ms/step - loss: 0.9060 - acc: 0.6220 - val_loss: 0.9670 - val_acc: 0.5500
Epoch 195/200
5/5 [==============================] - 0s 74ms/step - loss: 0.9056 - acc: 0.6300 - val_loss: 0.9672 - val_acc: 0.5000
Epoch 196/200
5/5 [==============================] - 0s 84ms/step - loss: 0.9014 - acc: 0.6080 - val_loss: 0.9661 - val_acc: 0.5357
Epoch 197/200
5/5 [==============================] - 0s 70ms/step - loss: 0.9230 - acc: 0.6060 - val_loss: 0.9797 - val_acc: 0.5143
Epoch 198/200
5/5 [==============================] - 0s 75ms/step - loss: 0.8983 - acc: 0.6360 - val_loss: 0.9679 - val_acc: 0.5429
Epoch 199/200
5/5 [==============================] - 0s 73ms/step - loss: 0.9098 - acc: 0.6060 - val_loss: 0.9778 - val_acc: 0.5214
Epoch 200/200
5/5 [==============================] - 0s 77ms/step - loss: 0.9115 - acc: 0.6020 - val_loss: 0.9701 - val_acc: 0.5429
```

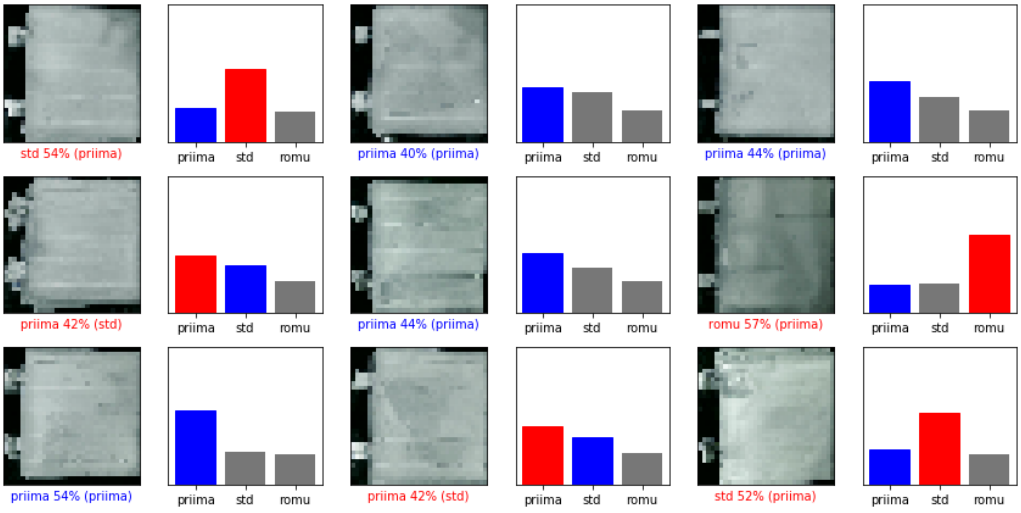
Kuva 34. *Opetusvaiheen tulostetta.*

Kun malli on valittu, malli opetetaan *model.fit(…)* komennolla, jolle annetaan parametrina opetuskuvat ja luokat, eräkoko, epokien määrä sekä testikuvat ja luokat. Viimeisien epokien tuloste opetusvaiheesta on esitetty Kuvassa 34. Kuvasta nähdään, että 200 epokin jälkeen, mallin tarkkuus opetusdatalle on 60 % ja testidatalle 54 %.

Opetustarkkuuksista piirrettiin myös kuvaajia. Yhden mallin opetuksen tarkkuudet on esitetty Kuvassa 35. Kuvaajassa nähdään opetusdatan oppimisen tarkkuus *(accuracy)* sekä tarkkuus, kuinka hyvin malli osaa luokitella testidatan *(validation accuracy)*. Kuvasta nähdään, kuinka tarkkuudet kasvavat hieman koko ajan epokien myötä. Kuvasta on nähtävissä myös, että tarkkuuksien ero alkaa suurentumaan eikä validointitarkkuus enää nouse. Tämä tarkoittaa, että malli oppii opetusdatan liian hyvin eikä osaa yleistää sitä tarpeeksi *(overfitting)*.



Kuva 35. *Opetuksen tarkkuudet.*



Kuva 36. *Arvioituja luokkia ja luokkien todennäköisyysjakaumat.*

Kuvassa 36 on esitetty yhdeksän ensimmäisen kuvan arvioitu luokka, arvioidun luokan todennäköisyys sekä suluissa oikea luokka. Väärät arviot on esitetty punaisella ja oikeat sinisellä. Kuvien vieressä on myös esitetty todennäköisyyksien jakauma eri luokille.

Myös *transfer learning* ja *fine-tuning* menetelmiä testattiin. *Transfer learning* tarkoittaa, että käytetään esiopetettuja painoja opettamiseen. Esiopetetussa verkossa vain viimeistä täysin yhdistettyä kerrosta eli luokittelukerrosta opetetaan. Jos esiopetettujen painojen luomisessa käytetyllä datakokoelmalla ei ole mitään tekemistä testeissä käytetyn datakokoelman välillä, voidaan käyttää *fine-tuningia. Fine tuning* tarkoittaa, että esiopetettujen painojen avulla tehdyn opetuksen jälkeen avataan kaikki kerrokset opettamiselle, jolloin saadaan malli toimimaan paremmin testeissä käytetyllä datakokoelmalla. [74]

Transfer learningia testattiin MobileNetillä [75] ja ResNetillä [76]. Se toteutettiin niin, että ensin ladattiin esiopetetut painot. Sen jälkeen kaikki paitsi viimeinen täysin yhdistetty kerros jäädytettiin, jotta niiden painoja ei muuteta opetuksen aikana. Tämän jälkeen malli opetettiin samalla tavalla kuin aiemmin. Kun tarpeeksi epokeja oli opetettu, tehtiin *fine-tuning* eli kaikkien kerrosten opettaminen otettiin käyttöön. *Transfer learningin* toteutus on esitetty Kuvassa 37. *Fine-tuning* tehtiin niin, että *pretrainedNet.trainable* asetettiin *Trueksi.*

```
pretrainedNet = applications.ResNet50(weights='imagenet', include_top=False, in-
put_shape=(32,32,3))
pretrainedNet.trainable = False
inputs = Input(shape=(32,32,3))
x = pretrainedNet(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
outputs = layers.Dense(3)(x)
model = Model(inputs, outputs)
```

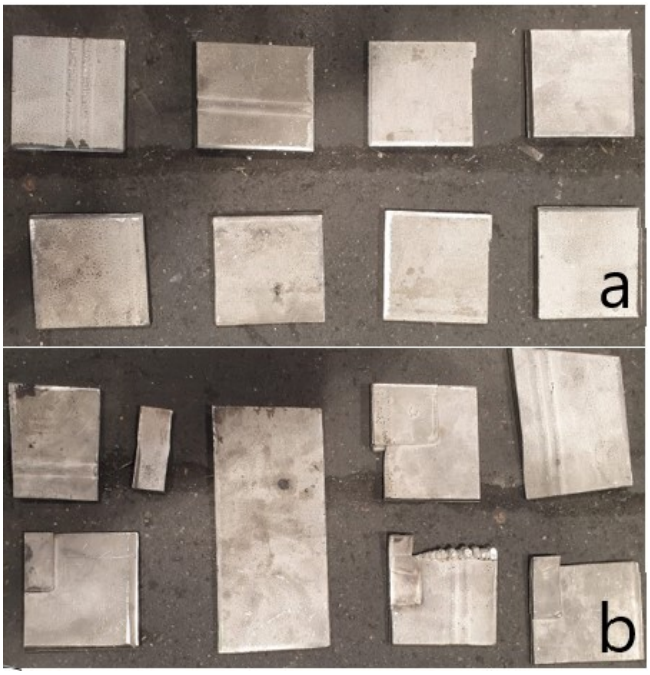
Kuva 37. *Transfer learningin toteutus.*

4.5 4-tuuman palan lajittelu

Samalla Labra.AI:n laitteistolla suoritettiin NNH:lla myös pienimuotoinen leikatun nikkelipalan testi. Testissä kokeiltiin liukuhihnalta tulevien 4-tuuman palojen lajittelua, jossa pyrittiin löytämään epäkurantit palat hyvien palojen joukosta. Epäkurantit palat ovat väärän kokoisia tai muotoisia paloja. Kuvasta 38 näkyy linjastolla liikkuvia 4-tuuman paloja.

Kuva 38. *4-tuuman paloja linjastolla.*

Kuvassa 39 on ylhäällä hyviä 4-tuuman paloja ja alhaalla on epäkurantteja paloja. Hyvät palat ovat selkeästi neliön muotoisia ja epäkurantit palat voivat olla liian isoja tai pieniä tai ne voivat olla leikattuna ”korvien” kohdalta. ”Korvat” ovat nikkelikatodeissa olevat pidikkeet, joiden avulla ne saadaan elektrolyysialtaisiin.



Kuva 39. *Lajiteltuja 4-tuuman paloja: hyviä 4-tuuman paloja (a) ja epäkurantteja 4-tuuman paloja (b).*

Heti testin alussa todettiin, että palat olivat niin limittäin, että kyseisellä laitteistolla oli mahdoton tehtävä erotella paloja toisistaan. Tästä syystä päädyttiin ratkaisuun, että ajetaan linjastolta yksittäisiä paloja ja luokitellaan niitä konenäön avulla.

Visionin ja kameran asetuksia piti muuttaa, koska palatestissä oli eri olosuhteet kuin katodien lajittelussa. Kuvat otettiin liikkuvista kappaleista liukuhihnalta. Valotusta jouduttiin pienentämään paljon ja sen seurauksena myös muita parametreja jouduttiin säätämään, koska valotus muutti kuvan kirkkautta. Myös kappaleen koko kuvassa oli eri kuin katodien lajittelussa ja näin ollen haetun kappaleen kokoparametreja säädettiin.

Kun parametrit oli saatu kohdilleen, opetus- ja testausvaiheet olivat samat kuin katodien lajittelussa: ensin mallille opetettiin hyviä 4-tuuman paloja sekä epäkurantteja paloja. Tämän jälkeen laitettiin malli päälle ja ajettiin erilaisia paloja linjalta. Testissä katsottiin, kuinka hyvin malli luokitteli palat. Jo pieni määrä opetettuja kuvia sai aikaan hyviä luokittelutuloksia. Testissä pystyttiin nopeasti toteamaan, että 4-tuuman palan lajittelu on huomattavasti helpompi kohde kuin katodien pinnanlaadun perusteella tehty lajittelu. Tähän syynä on se, että suurimpana kriteerinä lajittelussa oli palan koko eikä pinnanlaatu.

Haasteena 4-tuuman palojen lajittelutestissä olivat alussa sumeat kuvat, jotka johtuivat valotusparametrista. Näitä ei kuitenkaan saatu kokonaan pois, sillä välillä säätämisen jälkeen osa kuvista oli edelleen sumeita. Tämä olisi todennäköisesti saatu kokonaan pois, jos olisi käytetty enemmän aikaa parametrien säätöön. Toinen ongelma oli ohuet palat, joita ei pystynyt Visionilla tunnistamaan. Hyvän 4-tuuman palan pitää myös olla

tietyn paksuinen. Testi oli lyhyt, sillä jo pienillä opetusmäärillä saatiin hyvät tulokset ja voitiin todeta, että testi oli onnistunut ja konenäkö palojen lajittelussa voisi toimia hyvin. Samalla leikkurilla leikattiin muitakin palakokoja, mutta konenäöllä onnistuttiin lajittelemaan vain 4-tuuman paloja, koska muut palakoot menivät eri linjastolta. Tästäkin syystä testi jäi lyhyeksi.

5. KOKEELLISET TULOKSET

Tässä luvussa selostetaan ensin Visionilla saatuja tuloksia läpi, jonka jälkeen kerrotaan Tensorflow-kirjaston avulla tehtyjen luokittelutestien tuloksia. Näiden jälkeen käydään läpi molempien testien haasteita sekä niiden mahdollisia ratkaisuja.

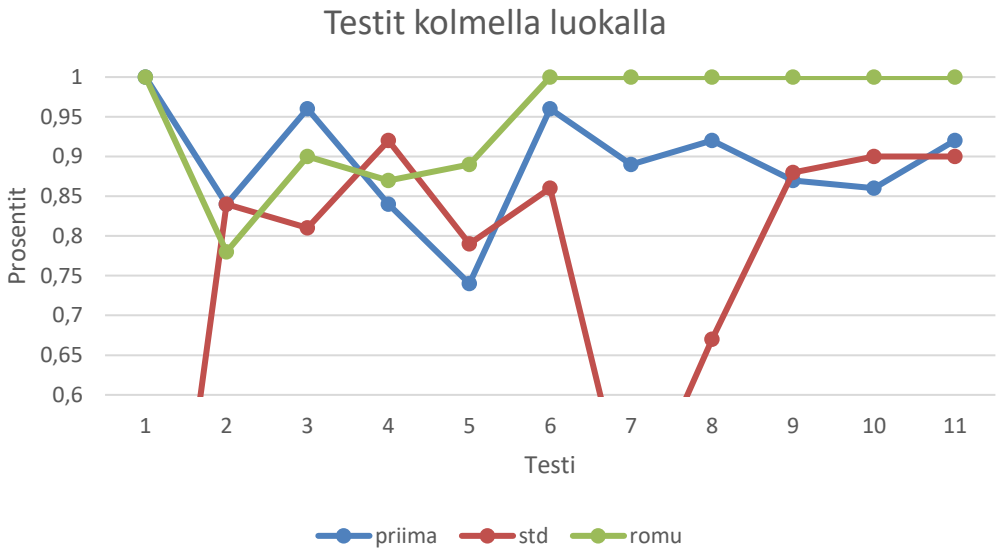
5.1 Visionilla saatuja tuloksia

Visionin mallin opetuksen jälkeen otettiin ylös montako opetettua kuvaa mistäkin luokasta on. Visionin mallia testattaessa pidettiin kirjaa, montako katodia on testattu ja monenko katodin kohdalla Vision luokitteli väärin sekä mistä laadusta ja miten luokittelu meni väärin. Lisäksi pidettiin kirjaa, minkälaisia virheet olivat. Esimerkiksi montako priimalaadun katodia Vision luokitteli standardiksi tai romuksi.

Jokaisen testin jälkeen tehtiin jotain muutoksia tai opetettiin lisää, jonka jälkeen aloitettiin uusi testi. Testien kirjanpito laajentui kokeiden edetessä. Ensimmäisten testien kirjanpito oli tästä syystä suppeampaa. Taulukossa 3 on esitetty viimeisimpien testien tuloksia (koko kirjanpito on esitetty liitteessä A). Opetetut katodit sarakkeessa ilmoitetut luvut tarkoittavat priimalaadun katodit + standardilaadun katodit + romulaadun katodit. Jokainen eri laadun sarake on jaettu kahtia. Näistä ensimmäisessä sarakkeessa on suhde oikein menneiden ja kokonaismäärän välillä ja toisessa sama luku on prosentteina.

Taulukko 3. *Lajittelutestien tuloksia. Testeistä nähtävissä opetusdatan koko sekä kokonaistunnistustarkkuus ja eri luokkien tunnistustarkkuudet.*

Testi	Opetetut katodit	Priima		Standardi		Romu		Oikeintunnistusprosentti	
1	155+104+9	1/1	100 %	2/22	9 %	2/2	100 %	5/25	20 %
2	169+170+40	57/76	84 %	113/135	84 %	25/32	78 %	195/243	80 %
3	220+232+71	71/74	96 %	46/57	81 %	19/21	90 %	136/152	89 %
4	223+243+82	37/44	84 %	86/93	92 %	33/38	87 %	156/175	89 %
5	250+250+114	45/61	74 %	50/63	79 %	8/9	89 %	103/133	78 %
6	272+265+113	78/81	96 %	37/43	86 %	10/10	100 %	125/134	93 %
7	270+282+118	26/29	89 %	4/9	44 %	1/1	100 %	31/39	79 %
8	270+282+118	33/36	92 %	4/6	67 %	1/1	100 %	38/43	88 %
9	270+282+118	98/113	87 %	30/34	88 %	4/4	100 %	132/151	87 %
10	280+294+121	128/148	86 %	110/122	90 %	2/2	100 %	238/270	88 %
11	300+314+124	59/64	92 %	36/40	90 %	0/0		95/104	91 %

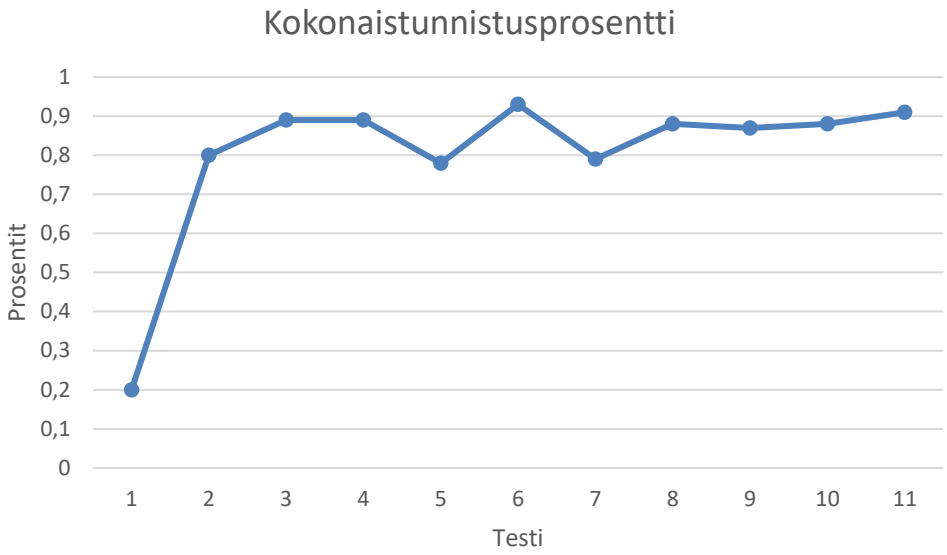


Kuva 40. *Visionin luokittelutestien luokkien tunnistusprosentit.*

Esimerkiksi Taulukon 3 testissä 2 priimalaatua tuli testin aikana 76 ja näistä Vision tunnisti 57 oikein, loput se luokitteli väärin joko standardiksi tai romuksi. Oikeintunnistusprosentti tarkoittaa montako katodia kokonaisuudessaan testin aikana on mennyt oikein. Oikeintunnistusprosenttisarake on jaettu samalla tavalla kuin yksittäiset laadutkin. Testeistä on myös havaittavissa, että opetettujen katodien määrät kasvavat koko ajan, mikä johtuu siitä, että testien välissä aina opetettiin lisää katodeja edellisten lisäksi, jotta malli saataisiin toimimaan paremmin. Testien testijoukot vaihtelevat paljon, mikä taas johtuu siitä, että välillä huomattiin nopeasti, että malli tarvitsee opettaa uudelleen.

Kuvasta 40 nähdään, miten eri laatujen tunnistusprosentit ovat muuttuneet testien aikana. Y-akselilta on poistettu kuvan selkeyttämiseksi 0,6 alle jäävät arvot, vaikka 2 arvoa jäi sen alapuolelle. Ensimmäinen arvo, joka on 0,6 alapuolella on ensimmäisessä testissä saatu standardilaadun tunnistusprosentti. Toinenkin arvo, joka on sen alapuolella, on standardilaadun tunnistusprosentti. Tämä johtui siitä, että Visioniin saatiin iso päivitys, joka muutti mallin opetusta. Tässä kohtaa jouduttiin taas opettamaan uusia katodeja, jotta malli saatiin toimimaan paremmin. Kuvasta nähdään myös, miten romulaadun tunnistus saatiin lopulta tasaantumaan 100 prosenttiin. Viimeisissä testeissä ei enää kerätty erikseen romukatodeja, koska todettiin, että Vision tunnistaa ne tarpeeksi hyvin. Tästä syystä romukatodeja oli viimeisissä testeissä vain vähän.

Kuvasta 41 nähdään kokonaistunnistusprosentit. Ensimmäisessä testissä kokonaistunnistusprosentti oli vain 20 %, mutta lopulta se saatiin tasaantumaan noin 90 prosenttiin. Lopussa joka testillä sitä saatiin kuitenkin myös parannettua. Kuvaajan notkahdus 5 testin aikana johtui siitä, että testin aikana siltanosturi oli valon päällä. Se siirrettiin pois, jolloin tulokset alkoivat taas parantua. Toinen notkahdus, joka tapahtui testin 7 aikana,



Kuva 41. *Visionin luokittelutestien oikeintunnistusprosentit.*

johtui edellä mainitusta ohjelman päivityksestä. Tästä myös huomataan, että notkahdus ei ole yhtä iso kuin standardilaadun tapauksessa. Tämä johtuu siitä, että standardilaatua ei saatu testin aikana tarpeeksi.

Visionin vääristä luokitteluista pidettiin myös kirjaa. Näin saatiin selville, minkälaisia virheet olivat. Kriittisimpiä virheitä olivat ne, jossa arvioitu luokka oli parempaa laatua kuin todellinen luokka. Näitä yritettiin minimoida. Taulukossa 4 on esitetty Visionin virheet. Kriittisten virheiden sarakkeet on merkitty taulukkoon harmaalla. Kuvasta nähdään, että kriittisistä virheistä ongelmana olivat virheet, jossa Vision luokitteli standardilaadun katodit priimalaaduksi. Tähän syynä on kuitenkin se, että standardi- ja priimalaadun rajatapaukset ovat niin lähellä toisiaan, että on vaikea sanoa, kumpaan luokkaan katodi oikeasti kuuluu. Tämä käy ilmi myös siitä, että virheitä, jossa Vision luokittelee priimalaadun standardiksi, on samassa suhteessa. Virhettä olisi saatu todennäköisesti vielä minimoitua opettamalla lisää rajatapauksia. Tässä on huomioitava kuitenkin se, että jo pieni määrä opetuksessa väärään luokkaan menneitä vaikuttaa toimivuuteen. Ja lisäksi luokitteluvirheiden tyyppiin ei voi vaikuttaa, mutta rajapintaa olisi voitu saada selkeämmäksi.

Visionin mallin opetuksen tuloksista validoinnin häviö, validoinnin tarkkuus ja opetuksen tarkkuus on listattuna Taulukossa 5. Nämä tulokset kertovat miten opetus on mennyt tunnetuilla kuvilla, mutta eivät kerro miten malli itsessään toimii, kun sille annetaan tuntemattomia kuvia. Taulukosta nähdään, miten testidatan häviö ja luokittelutarkkuus heikkeni testien myötä. Tämä johtuu todennäköisesti siitä, että joka testissä oli hieman enemmän dataa, jolloin malli ei enää oppinut yhtä hyvin samalla määrällä epokeja. Molempia

Taulukko 4. *Luokitteluvirheiden tyypit.*

Testi	priima -> std	priima -> romu	std -> romu	std -> priima	romu -> priima	romu -> std
4	7	0	1	6	1	4
5	13	3	8	5	0	1
6	3	0	0	6	0	0
7	3	0	0	5	0	0
8	3	0	0	2	0	0
9	15	0	0	4	0	0
10	20	0	0	12	0	0
11	5	0	0	4	0	0

Taulukko 5. *Visionin testien opetuksien tuloksia.*

Testi	Validation loss	Validation accuracy	Train accuracy
4	0,39	0,87	0,99
5	0,21	0,94	0,98
6	0,26	0,9	0,98
7	0,7	0,6	0,91
8	0,68	0,61	0,91
9	0,68	0,66	0,86
10	0,69	0,69	0,9
11	0,79	0,63	0,9

olisi todennäköisesti saatu paremmaksi, jos olisi opetettu useamman kerran eli enemmän epokeja. Tämä olisi myös voinut olla järkevää testien kannalta.

5.2 Tensorflowlla tehtyjen testien tuloksia

Tensorflowlla testattiin LeNetiä, kustomoitua neuroverkkomallia sekä *transfer learningia* ja *fine-tuningia*. Saadut testidatan tunnistamistarkkuudet olivat suurimmaksi osaksi väliltä 40-70 %. Paras tulos ilman esiopetettuja painoja oli kustomoidulla mallilla saatu 74 %. Esiopetettujen painojen ja ResNetin avulla päästiin helposti yli 80 % ja lopulta paras tulos oli 90 %. Testeissä vaihdeltiin aina yhtä arvoa kerrallaan ja pyrittiin pääsemään parhaaseen lopputulokseen. Kuvat olisi voitu muuttaa värikuvista harmaiksi ja optimointifunktiona olisi voitu kokeilla muitakin menetelmiä kuin *'adam'*ia, mutta näitä ei testattu.

Ensin testattiin LeNet mallia, jonka parhaimmat tulokset näkyvät Taulukossa 6. LeNetiä kustomoitiin sekä kokeiltiin eri tavoilla. Oppimisasteena parhaiten toimi 0,001. Malli toimi parhaiten, kun eräkoko oli puolet datan määrästä. *Poolingina* paras oli *average* ja aktivointifunktiona *'tanh'*. LeNetillä paras tarkkuus oli 70 %, joka saatiin, kun data oli laajennettu ja augmentoitu pelkillä *flipeillä.* LeNetillä tehdyissä testeissä opetettiin 500 epokia.

Taulukko 6. *LeNet testien tuloksia.*

Testi	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Erä	500	500	250	250	250	500	1000	500
Oppimisaste	0,00001	0,001	0,001	0,001	0,001	0,001	0,001	0,001
Vertical flip	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Horizontal flip	No	Yes	Yes	Yes	Yes	Yes	Yes	No
Kirkkaus	No	No	No	Yes	No	No	No	Yes
Zoom	No	No	No	No	Yes	No	No	Yes
Siirtymä	No	No	No	No	No	No	No	Yes
Laajennettu data	No	No	No	No	No	Yes	Yes	Yes
Pooling	avg	avg	avg	avg	avg	avg	avg	avg
Aktivointi	tanh	tanh	tanh	tanh	tanh	tanh	tanh	tanh
Tarkkuus	**0,6**	**0,61**	**0,64**	**0,62**	**0,62**	**0,7**	**0,65**	**0,62**

Taulukko 7. *Kustomoidun mallin tuloksia.*

Testi	**1**	**2**	**3**	**4**	**5**	**6**
Pooling	max	average	max	max	max	max
Filtterit	32	32	64	16	16	16
Filtterikoko	3	3	3	3	2	3
Dense ker-roksia	100	100	100	100	100	1000
Tarkkuus	**0,64**	**0,57**	**0,4286**	**0,74**	**0,72**	**0,7**

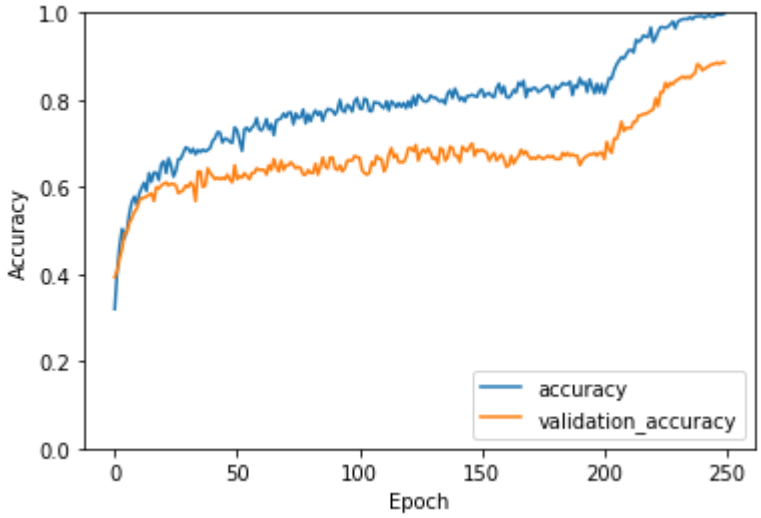
LeNetin jälkeen testattiin kustomoitua mallia, jonka tulokset näkyvät Taulukossa 7. Kustomoitua mallia testattiin jo ennen LeNetiäkin, mutta tulokset saatiin järkeviksi vasta, kun LeNet oli saatu toimimaan hyvin ja siinä saatuja havaintoja käytettiin avuksi. Taulukossa on esitetty vain ne parametrit, jotka eroavat toisistaan. Tosiasiassa säädettyjä parametreja oli enemmän. Kaikissa on käytetty laajennettua dataa. Aktivointifunktiona kaikissa oli *'tanh'*, eräkoko oli 500 ja epokeja 200. Kustomoitua mallia olisi voinut testata vielä paljon enemmän erilaisilla kombinaatioilla.

Viimeisenä testattiin *transfer learningia* ja *fine-tuningia*, jonka parhaimmat tulokset on esitetty Taulukossa 8. Näiden testien aikana käytettiin hyväksi edellisissä testeissä havaittuja asioita. Tästä syystä käytettiin laajennettua dataa, eräkokoa 500 ja aktivointifunktiona *'tanh'*ia. Testeissä kokeiltiin MobileNetiä ja Resnetiä. Taulukossa näkyvissä testeissä on kaikissa käytetty *flippejä* ja kirkkauden augmentointia, joten niitä ei ole lisätty taulukkoon.

Taulukko 8. *Transfer learningin ja fine-tuningin tuloksia.*

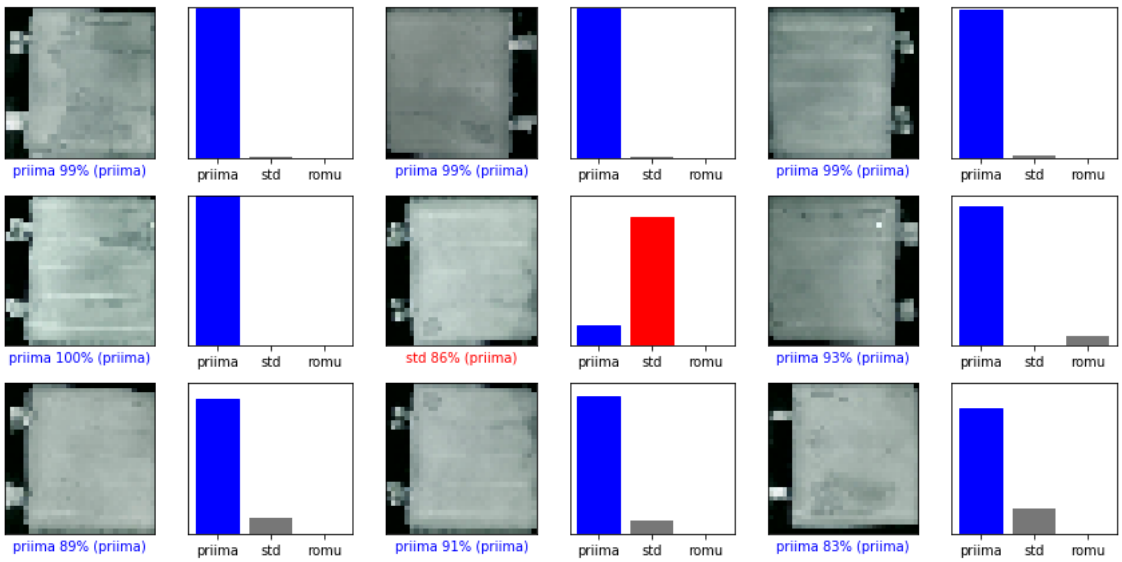
Testi	**1**		**2**		**3**		**4**	
Kuvakoko	32		32		64		32	
Zoom	Yes		Yes		Yes		No	
Verkko	MobileNet		ResNet		ResNet		ResNet	
Opetus	transfer	fine-tune	transfer	fine-tune	transfer	fine-tune	transfer	fine-tune
Epokit	200	30	200	20	200	20	150	50
Oppimis-aste	0,001	0,001	0,001	0,001	0,001	0,001	0,001	0,00001
Tarkkuus	**0,49**	**0,6**	**0,73**	**0,82**	**0,78**	**0,83**	**0,68**	**0,9**

Transfer learningin ja *fine-tuningin* avulla saatu paras tarkkuus oli 90 %. Kuvassa 42 on esitetty *transfer learningin* ja *fine-tuningin* opetuksen tarkkuudet. Kuvasta nähdään kuinka 200 epokin kohdalla on siirrytty *fine-tuningiin* ja tarkkuudet nousevat nopeasti. *Transfer learningin* avulla saatu opetustarkkuus oli noin 80 % ja validointitarkkuus noin 65 %. *Fine-tuningin* jälkeen ne opetustarkkuus nousi lähelle 100 % ja validointitarkkuus noin 90 %.



Kuva 42. *Transfer learningin ja fine-tuningin opetuksen tarkkuudet.*

Kuva yhdeksästä ensimmäisestä kuvasta datassa ja niiden arvioidut luokat sekä todennäköisyysjakaumat on esitetty Kuvassa 43. Väärät arviot on esitetty punaisella ja oikeat sinisellä. Kuvasta nähdään, kuinka selkeästi monet luokat on tunnistettu. Toisaalta yksi kuva on tunnistettu väärin standardilaaduksi, vaikka kuva on priimalaatua. Malli on 86 % varma, että kuva on standardilaatua. Tämä on todennäköisesti rajatapaus, joka voi olla kumpaa laatua vain.



Kuva 43. *Arvioituja luokkia ja luokkien todennäköisyysjakaumat.*

5.3 Haasteita

Heti kokeellisen osuuden alkuun ohjelmassa havaittiin ohjelmointivirheitä. Myös myöhemmissä vaiheissa ilmeni muutamia virheitä. Kaikki virheet saatiin korjattua yhteistyössä Labra.AI:n kanssa. Ohjelmaan saatiin myös muutama isompi päivitys, jotka paransivat toimivuutta ja toivat lisää ominaisuuksia.

Suurimpana ongelmana testien aikana oli valaistus ja sen muutokset. Pienetkin valaistuksen muutokset testialueella vaikuttivat heti luokittelutuloksiin. Erään testin aikana kameran päältä katosta vaihdettiin lamppu, jolloin alue kirkastui. Tällöin jouduttiin säätämään Visionin parametreja valaistuksen muututtua. Katossa olevan siltanosturin ollessa valon edessä, kamerasta tuleva kuva oli erittäin pimeä, jolloin jouduttiin odottamaan, että nosturi saatiin siirrettyä pois tieltä. Yhtenä ongelmana oli myös auringon valo. Tiettyyn kellonaikaan se paistoi suoraan katodin päälle, jolloin kuva näytti kirkkaammalta. Hallin ikkunoissa ei ollut verhoja, joten ainoa ratkaisu oli odottaa auringon valon siirtymistä tai säätää parametreja.

Kuvan parametrien säädössä ongelmana oli erittäin kiiltävä katodin pinta, joka oli kuin peili. Näin ollen kirkkauteen vaikuttavat parametrit piti säätää pieniksi. Parametrien säätöä vaikeutti lisää se, että piti saada tarkka kuva, jotta siitä nähtiin, mitä laatua katodi on. Osassa katodeista on kuparin värisiä ohuita viivoja pinnassa, joita ei priimalaadun katodeissa saisi olla yhtään. Kupariraitoja ei saatu kameran parametreja säätämälläkään näkyviin, joten katodien lajittelun toimivuutta testattiin ilman sitä. Kupariraidan ja pinnan näkyvyyteen olisi voinut auttaa, jos kameraa olisi tuotu alemmas, jolloin kuvaa ei olisi

tarvinnut zoomata ja kuvata niin kaukaa. Manuaaliselle lajittelupisteelle rakennettua kameran telinettä olisi voitu lyhentää noin metrin verran, niin että olisi pystytty vielä työskentelemään, mutta tätä ei testattu.

Kuvasta oli myös hankala erottaa pieniä dendriittejä, joita oli joidenkin katodien pinnoissa. Jos pieniä dendriittejä on paljon, katodi pitäisi tunnistaa standardilaadun katodiksi, mutta usein, kun Vision ei erottanut niitä, se luokitteli katodin priimaksi. Kameran parametrien säätämiseen käytettiin todella paljon aikaa, mutta täydellistä kuvaa ei saatu. Parametreja myös muutettiin usein myöhemmin.

Kokeellisen osuuden aikana mietittiin myös kameran etäisyyttä, että olisiko parempi, jos se olisi lähempänä kuvattavaa nippua. Tätä ei kuitenkaan lähdetty muuttamaan, vaan todettiin, että saatiin tarpeeksi hyvät tulokset kauempaakin. Kokeellisen osuuden aikana lajittelupisteen vieressä suoritettiin myös leikkausta, jolloin kamera liikkui vähän ensimmäisellä kerralla. Todettiin kuitenkin, että kamera oli huonosti kiristetty, sillä kiristyksen jälkeen ongelmaa ei enää ollut.

Katodiniput pestään ennen kuin ne tuodaan lajiteltavaksi. Jos nippu on tuore, niin se on vielä märkä, kun se tulee lajiteltavaksi. Tämä vaikuttaa taas luokitteluun, koska osa katodeista on märkiä ja osa ei. Märät katodit heijastavat valoa kuvissa. Tästä ongelmasta päästiin eroon, kun opetettiin ohjelmaa suurella määrällä kuivia sekä märkiä katodeja. Ideaalitilanteessa kaikki katodit olisivat kuivia.

Katodien paksut sekä ohuet reunat, mustunut yläreuna ja taittuneet kulmat olivat ongelmia, joita oli vaikea havaita Visionilla. Jos katodilla on liian paksut reunat se ei ole priimalaatua ja tätä ei Visionilla pysty näkemään. Myöskään ohutta reunaa ei pysty näkemään. Jos katodin yläreuna on mustunut vähän, se ei ole priimaa. Musta yläreuna aiheutuu siitä, jos elektrolyysissä altaan liuospinta on ollut liian matalalla ja koko katodi ei ole ollut pinnan alla. Näitä tapauksia oli testin aikana vain vähän ja tämä olisi pystytty opettamaan Visionille, jos mustuneita katodeja olisi ollut enemmän. Myös taittuneita kulmia oli vain vähän ja tämä olisi pystytty opettamaan, jos niitä olisi ollut tarpeeksi.

Välillä niput olivat sellaisia, että päältäpäin kuvattuna päällimmäisen katodin alta näkyi muita katodeja. Tämä tuotti ongelmia varsinkin, jos alapuolella oleva katodi oli palanut ja päällimmäinen katodi oli hyvä. Vision saattoi luokitella katodin romuksi. Tämä saatiin korjattua niin, että siirrettiin päällimmäistä katodia niin, että toista katodia ei enää näkynyt ja otettiin uusi kuva.

Osassa kuvista näkyi imukuppinostimen varjo eli kuvat oli otettu liian nopeasti. Tätä ei huomattu heti ja malliin oli opetettu sellaisiakin kuvia paljon, joten sillä ei todennäköisesti

ollut suurta vaikutusta. Kuitenkin välillä testattaessa Vision luokitteli jostain selvästä tietyn laadun katodista, jotain aivan muuta kuin olisi pitänyt, mutta kun otettiin uusi kuva se osasikin luokitella sen oikein.

Vision ottaa kuvan aina, kun se tunnistaa uuden staattisen kappaleen, joka on oikean kokoinen. Tämä aiheutti paljon ongelmia, mutta parametreja säätämällä tätäkin saatiin paremmaksi. Alussa Vision otti todella paljon kuvia samasta kappaleesta. Tähän vaikutti esimerkiksi nipun heilunta. Vision otti myös kuvia, kun imukuppinostin oli edessä eli se tunnisti imukuppinostimenkin samaksi kappaleeksi. Lisäksi ohjelma saattoi ottaa monta kuvaa, jos esimerkiksi imukuppinostimen vaijeri heilui. Myös nostimen ja vaijerin varjot vaikuttivat kuvien ottamiseen. Kun katodia siirrettiin pois, varjot ja kuva muuttuivat, jolloin Vision otti monta kuvaa. Kuvien ottamista saatiin pienemmäksi nostamalla etualan kynnysarvoa.

Taustan opettamisessa oli myös ongelmia. Aluksi yritettiin opettaa tausta, kun trukkilavat, jonka päälle katodiniput tuotiin, olivat taustassa mukana. Trukkilava ja katodi näyttivät kuvassa kuitenkin liian saman värisiltä. Kun katodinippu oli asetettu paikalleen, ohjelma luuli, että trukkilavan paikalla mikään ei ole muuttunut ja kuvitteli kappaleen siis eri muotoiseksi. Kuva 44 havainnollistaa asiaa. Tämä ongelma olisi voitu todennäköisesti ratkaista parametreja säätämällä, mutta siihen ei kulutettu aikaa ja tausta opetettiin jatkossa ilman trukkilavoja.

Myös tunnistimen asetukset tuottivat ongelmia. Aluksi etualan kynnysarvo oli melko pieni ja varjojen tunnistus oli päällä. Tämä kuitenkin tuotti välillä kappaleen tunnistusongelmia, jonka vuoksi varjojen tunnistus otettiin pois päältä ja etualan kynnysarvoa nostettiin toiseen ääripäähän.

Priimalaadun ja standardilaadun ero tuotti myös haasteita, koska priimalaadun ja standardilaadun ero on aika ajoin vaikea määritellä. Tästä syystä osaa Visionin virheistä ei merkattu ylös. Esimerkiksi kirjanpitoon ei merkattu välillä mitään, jos katodi oli priimaa ja Vision luokitteli sen standardiksi, sillä katodin olisi voinut luokitella kumpaan vaan luokkaan.

Romukatodien saatavuus oli myös ongelma, koska romua tulee elektrolyysistä huomattavasti vähemmän kuin hyvää katodia. Tästä syystä kerättiin sivuun muutamia romunippuja, joista osa opetettiin Visionille ja osaa käytettiin testaukseen. Romukatodien ja hyvien katodien raja on kuitenkin selvä verrattuna priimalaadun ja standardilaadun rajaan, joten romukatodeja ei tarvittu niin paljon.

Osa romukatodeista oli niin palaneita ja mustia, että Vision ei tunnistanut mustaa aluetta samaksi kappaleeksi. Tästä syystä Visionin näkemä kappale oli pienempi kuin haettu

Kuva 44. *Taustan opettamisongelma, jossa trukkilava on liian samanvärinen kuin katodi. Kuvassa näkyy trukkilavan varjo katodin päällä, sillä Vision kuvittelee trukkilavan olevan vieläkin kuvassa.*

kappaleen koko ja Vision ei tunnistanut kappaletta. Jos katodi täytti kuitenkin minimikappalerajat, Vision otti siitä kuvan. Tämäkin olisi todennäköisesti saatu säädettyä parametrien avulla.

Tensorflowlla tehdyissä testeissä haastavaa oli löytää oikeita parametreja malleille. Kustomoidun mallin rakentaminen oli myös haastavaa, sillä vaihtoehtoja ja parametreja, joita muuttaa, on niin paljon. Testejä tehtiin paljon ja välillä niin nopeasti, ettei kaikkea pystynyt kirjaamaan ylös. Tärkeimmät oivallukset ja tulokset kuitenkin saatiin ylös. Osa testiajoista oli aikaa vieviä ja tästä syystä välillä tyydyttiin esimerkiksi pienempään kuvakokoon, vaikka isommalla kuvakoolla olisi voitu saada parempia tuloksia. Monet testeistä olivat kokonaan turhia ja tarkkuus jämähti tai pysyi muuttumattomana 0,4286 arvossa. Yksi kriittinen tekijä oli myös datan määrä. Parempia tuloksia alettiin saada, kun dataa laajennettiin augmentoimalla. Olisi ollut siis hyvä, että dataa olisi ollut enemmän, sillä liika augmentonti voi johtaa *overfittingiin*. *Fine-tuning* testien alussa, ennen oikeiden parametrien löytämistä, testidatan tarkkuus romahti nopeasti *transfer learningin* aikana, vaikka opetusdatan tarkkuus nousi. Tällöin *fine-tuningistakaan* ei ollut hyötyä.

6. KONENÄÖN SOVELTUVUUS LAJITTELUUN

Tässä luvussa vertaillaan ensin kaupallisen ratkaisun sekä itse tehtyjen ohjelmointien tuloksia. Tämän jälkeen tehdään päätelmiä ja mietitään konenäön soveltuvuutta nikkelikatodien lajitteluun.

6.1 Vertailu

Vision-ohjelmistolla päästiin hieman parempiin tuloksiin kuin Pythonin Tensorflowlla testatuilla konvoluutioneuroverkkomalleilla. Visionilla paras tarkkuus oli 93 %. Ilman esiopetettuja painoja kustomoidulla verkolla tarkkuudeksi saatiin 74 %. Esiopetettujen painojen avulla päästiin yli 80 % ja parhaimmillaan tarkkuudeksi saatiin 90 %. Visionilla tarkkuudet olivat huonoimmillaankin yli 70 %, kun taas kustomoiduilla verkoilla tarkkuudet olivat suurimmaksi osaksi 40-70 % välillä. Visionilla tehtyjen testien keskiarvo oli 81 %. Siihen on monta syytä, miksi kustomoiduilla verkoilla ei päästy samoihin tarkkuuksiin. Syitä on käsitelty seuraavaksi.

Konvoluutioneuroverkolle oli vaikea löytää sopivaa mallia, jolla olisi saatu hyviä tuloksia. Visionilla tätä ei tarvinnut miettiä vaan malli oli valmiina. Konvoluutioneuroverkkojen mallien etsimiseen ja testailuun sekä mallien parametrien säätöön meni paljon aikaa.

Augmentointimenetelminä kustomoiduille verkoille käytettiin enimmäkseen *vertical ja horizontal flippejä*. Muutamaa muutakin menetelmää kokeiltiin, mutta suurin osa vain heikensi tarkkuuksia. Visionilla augmentointia päästiin säätämään vasta kokeellisen osuuden lopussa ison päivityksen myötä. Aluksi augmentointi tapahtui piilotetusti opetuksen aikana. Tästä syystä Visionin erilaisia augmentointi-mahdollisuuksia ei kokeiltu paljoa.

Visionin piilotetussa ohjelmassa on paljon kuvien segmentointia ja esikäsittelyä. Näitä ei tehty lainkaan kustomoiduille verkoille vaan tyydyttiin Visionilla otettuihin kuviin ja syötettiin niitä malleille. Näiden kuvien joukosta poistettiin epäselviä kuvia sekä luokkien rajatapauksia. Tätä tehtiin myös Visionin mallin kuville, mutta ei niin paljoa.

Molemmissa tapauksissa mallit olisivat todennäköisesti toimineet paremmin, jos dataa olisi kerätty enemmän. Esimerkiksi Cifar10 datakokoelma sisältää 60 000 kuvaa, kun tässä oli kerättynä alle 1000 kuvaa [77]. Kustomoiduilla verkoilla alettiin saada parempia tuloksia, kun laajennettiin dataa augmentoimalla. Visionilla meni paljon aikaa kuvan parametrien säätöön ja tästä syystä dataa ei saatu kerättyä enempää. Visionin tulokset olivat myös sen verran hyviä, että siihen ei tarvittu enempää kuvia, vaikka kustomoiduille verkoille olisi vielä tarvittu.

Visionilla mallin opettamiseen käytettiin esiopetettuja painoja, joka auttoi niiden optimoimista. Myös kustomoiduilla verkoilla esiopetettujen painojen käyttö auttoi saamaan paljon parempia tuloksia, mutta se myös kasvatti opetukseen menevää aikaa.

Molemmissa testeissä olisi voinut olla paremmat olosuhteet ja lähtökohdat. Jos Visionin testeissä olisi ollut optimivalaistu kuvauskoppi, parametrien säätöön ei ehkä olisi mennyt niin paljon aikaa. Kustomoiduilla verkoilla paremmat tulokset olisi voitu saada, jos kuville olisi tehty esikäsittelyä ja käytetty enemmän augmentointia.

6.2 Päätelmät

Visionin luokittelutesteistä saatiin hyviä tuloksia ja viimeisessä testissä Visionille oli opetettuna 750 eri kuvaa ja melkein joka testissä testattujen kuvien määrä oli yli 100. Tarkkuudeksi saatiin parhaimmillaan 93 %. Kaikkien Visionilla tehtyjen testien keskiarvo oli 81 % ja viimeisten 10 testin keskiarvo oli 87 %. Kustomoiduilla konvoluutioneuroverkoilla tarkkuudet olivat suurimmaksi osaksi väliltä 40-70 %, mutta lopulta esiopetettujen painojen avulla päästiin myös 90 %. Konenäkö soveltuu nikkelikatodien luokitteluun näiden perusteella hyvin. Vielä parempiin tuloksiin olisi todennäköisesti päästy, jos opetettujen katodien määrää olisi kasvatettu vielä. Mahdollisen linjaston konenäkömallille olisi hyvä opettaa vähintään 500 priimalaadun sekä 500 standardilaadun levyä, jotta mallista saadaan varmasti toimiva.

Romulaadun nikkelikatodeja tulee vähemmän kuin muita laatuja. Romut kuitenkin eroavat pinnanlaadultaan niin paljon muista, että niitä ei tarvitse opettaa yhtä paljon. Testeissä jo 100:lla romukatodin kuvalla päästiin 100 % tarkkuuteen romujen tunnistuksessa. Sen sijaan priimalaadun ja standardilaadun katodeja olisi voinut opettaa lisää, jotta niiden välistä tunnistusta olisi saatu vielä paremmaksi. Epäselvissä tapauksissa tärkeää on, että konenäkö ei luokittelisi standardilaadun katodeja priimalaaduksi. Jos konenäkö arvioi eri luokille todennäköisyydet, priimalaadulle voisi asettaa raja-arvon, että esimerkiksi todennäköisyys pitää olla vähintään 75 % ennen kuin konenäkö luokittelee katodin priimalaaduksi. Toisaalta tällä ratkaisulla moni oikeasti priimalaadun katodikin jäisi tunnistamatta, jos tarkkuus ei riitä.

Jos elektrolyysin prosessissa jokin muuttuu ja sen seurauksena nikkelikatodien pinnat muuttuvat, tämä pystytään opettamaan mallille lisäopetuksella. Esimerkiksi nikkelikatodien siemenlevyjen muuttunut kuvio saadaan todennäköisesti opetettua mallille. Lisäopettamiselle on siis oltava mahdollisuus. Jossain tapauksissa voidaan haluta myös tehdä kokonaan uusi malli tai monta mallia.

Visionilla ei pystytty tunnistamaan katodien painoja, eikä niiden käyryyttä. Lajittelulinjastoon voisi lisätä vaa'an niin, että yksittäinen katodi saadaan punnittua. Jos katodi on liian kevyt tai painava, sen voisi luokitella romuksi. Katodin käyryys voitaisiin selvittää, jos ne kuvataan sivusta sekä päältä ja jollain konenäön menetelmällä todennäköisesti saataisiin tarkasteltua katodin suoruutta. Yksittäisen katodin painon ja käyryyden selvittäminen tukevat sitä ajatusta, että katodit olisi kuvattava yksitellen. Lisäksi se, että katodit ovat nipussa saattavat aiheuttaa ongelmia, sillä pinon alta saattaa näkyä katodeita, jotka vääristävät kuvaa ja vaikuttavat luokitteluun.

Visionin testeissä osassa nikkelikatodeista olevia kupariraitoja ei saatu näkyviin. Tähän olisi voinut auttaa, jos kamera olisi tuotu lähemmäs ja jos kameran parametreja olisi säädetty vieläkin enemmän. Parempia tuloksia olisi todennäköisesti myös saatu, jos kaikki katodit olisivat olleet kuivia. Märät katoditkin saatiin opetettua hyvin, vaikka vesipisarat katodin pinnalla heijastivat välillä kuvassa, mikä vaikeutti luokittelua.

Tärkein huomio on kuitenkin valaistuksen merkitys. Visionin testeissä, valaistuksen muuttuessa, luokittelutulokset muuttuivat huonommiksi. Tällöin joko opetettiin lisää tai säädettiin parametreja. Tämä ei ollut hyvä asia, koska kun valaistus muuttui taas normaaliksi, parametreja jouduttiin taas hienosäätämään. Lajittelulinjastossa on oltava oma vakiovalaistus ja mahdollisesti jopa oma optimivalaistu huone, jossa kuvaus tapahtuu. Muun, esimerkiksi auringosta tulevan valon, pääsy lajittelulinjastolle tulisi estää.

7. LAJITTELULINJASTON ESISUUNNITTELU

Tässä luvussa suunnitellaan NNH:n katodien lajitteluun automaattista lajittelulinjastoa. Luvussa 7.1 käydään läpi lajittelulinjastolla huomioitavia asioita. Luvussa 7.2 esitetään muutama erilainen malli mahdolliselle lajittelulinjastolle.

7.1 Huomioon otettavia asioita

Lajittelulinjaston suunnittelussa pitää huomioida monia asioita. Huomioitavia asioita on sekä ohjelmallisia että rakenteellisia. Luvussa käsitellään katodin kuvaamista, robottiin liittyviä asioita sekä lopussa muita huomioitavia asioita.

Lajittelutesteissä katodit kuvattiin vain toiselta puolelta. Lajittelulinjastossa pitää katodista ottaa kuva molemmilta puolilta, jotta sille saadaan määritettyä oikea laatuluokka. Priimalaadun katodin pitää olla priimalaatua molemmilta puolilta. Kuvaamisessa pitää olla optimivalaistus, jotta valaistus olisi aina sama kuvia otettaessa. Tähän ratkaisuna voisi olla ison avoimen tilan sijaan pienempi valaistu kuvauskoppi, jossa kuvat otettaisiin. Kuvat pitää myös ottaa oikeaan aikaan, jottei esimerkiksi robotin varjo näy kuvassa. Testeissä todettiin, että välillä nipun alemmat katodit näkyivät liikaa kuvissa ja mahdollisesti vaikuttivat lopputulokseen. Tästä syystä voisi olla hyvä, että lajittelulinjastossa katodit kuvattaisiin yksi kerrallaan. Lisäksi kuten luvussa 6.2 todettiin, kuvattavien katodien olisi hyvä olla kuivia, jos mahdollista.

Katodin käyryyttä, paksuutta ja painoa ei saada määritettyä, jos kuvataan vain edestäpäin. Jos linjastolla kuvataan vain yksi katodi kerrallaan, kuvauspisteellä voisi olla vaaka, jolloin saataisiin myös yksittäisen katodin paino selville. Käyryys voitaisiin saada selville, jos katodi kuvataan myös ylhäältä sekä sivusta. Tähän tarvittaisiin pari kameraa lisää tai vaihtoehtoisesti kamera voisi olla esimerkiksi robotin päässä. Näin kuvaamiseen menisi kuitenkin enemmän aikaa, sillä robotti joutuisi liikkumaan ja ottamaan monta kuvaa useasta paikasta.

Linjastoon tarvittavien robottien riittävä ulottuvuus pitää ottaa huomioon, jotta ne ylettyvät tarvittaviin paikkoihin. Lisäksi roboteilla pitää olla riittävä hyötykuorma, sillä nikkelikatodit painavat tyypillisesti 60–80 kg, mutta paksuimmat jopa 100 kg. Tästä syystä myös robottien tarttujien pitää olla oikeanlaiset ja riittävän vahvat, etteivät ne pudottele katodeja. Jotta linjasto saadaan toimimaan mahdollisimman optimaalisesti ja nopeasti, myös robottien määrä on huomioitava. Robottien liikeradat on huomioitava, jotta robotit pääsevät

liikkumaan hyvin. Linjastosta pitää tehdä tarpeeksi laaja, jotta robottien ei tarvitse liikkua liian pienessä tilassa.

Linjastolla olisi hyvä olla yksi yhteinen ohjauspaneeli, josta robotteja, kuljettimia, sitomakonetta sekä muita toimintoja saisi ohjattua. Jos jokaisella linjaston laitteella on oma ohjauspaneeli, se monimutkaistaisi linjaston käyttöä huomattavasti. Linjastolla pitää olla myös mahdollisuus manuaaliseen luokitteluun esimerkiksi lajittelussa esiintyvän vian, kuten kameran rikkoutumisen sattuessa. Operaattori pystyisi tällöin määrittämään ohjauspaneelista katodien luokat, jolloin robotti osaisi viedä katodin oikeaan paikkaan. Sama ominaisuus tarvitaan myös tilanteisiin, jossa luokittelumallia tarvitsee opettaa lisää tai sille kerrotaan, että arvioitu luokka on väärä.

Lajiteltavien luokkien määrä vaikuttaa kuljettimien määrään ja siksi niiden määrä on mietittävä tarkkaan. Tällä hetkellä luokat ovat priima, standardi ja romu, mutta jos halutaan esimerkiksi leikattava luokka, kuljettimia pitää olla enemmän. Ainakin priimalaadun ja standardilaadun niput pitää sitoa teräsvanteilla. Pitää myös miettiä, onko kuljettimilla yhteinen sitomakone vai molemmilla omat. Romuniput sidotaan eri tavalla ja niitä valmistuu huomattavasti vähemmän kuin muita nippuja, joten näiden sitominen voidaan hoitaa manuaalisesti.

Kuljettimien on syytä olla myös tarpeeksi pitkät, jotta niille mahtuu monta nippua samaan aikaan. Näin automaattinen linjasto jatkaa toimintaa nipun tullessa valmiiksi, vaikka sitä ei heti nostettaisi linjalta pois. On huomioitava myös lajittelun nopeus. Kuinka nopeasti linjasto lajittelee yhden katodin? Entä yhden nipun? Kuljetin, johon trukki tuo nippuja lajiteltavaksi voi täyttyä ja nippuja joudutaan viemään muualle, jos lajittelu on liian hidasta. Joka tapauksessa olisi hyvä ajatella jokin paikka, johon nippuja varastoidaan, jos kuljettimelle ei mahdu.

Linjaston ympärille on asennettava tarpeeksi valoverhoja ja turvalaitteistoja, mutta kuitenkin niin, että huomioidaan myös ongelmatilanteet. Jos katodeja tippuu linjastolle, miten ne saadaan pois? Katossa voisi olla nosturi, joka ei olisi tiellä ja ylettyisi mahdollisimman isolle alueelle. Olisi hyvä, jos operaattorin tarvitsisi nostella painavia nikkelikatodeja mahdollisimman vähän. Linjastolle pitää siis myös päästä mahdollisimman helposti turvalaitteistoista huolimatta.

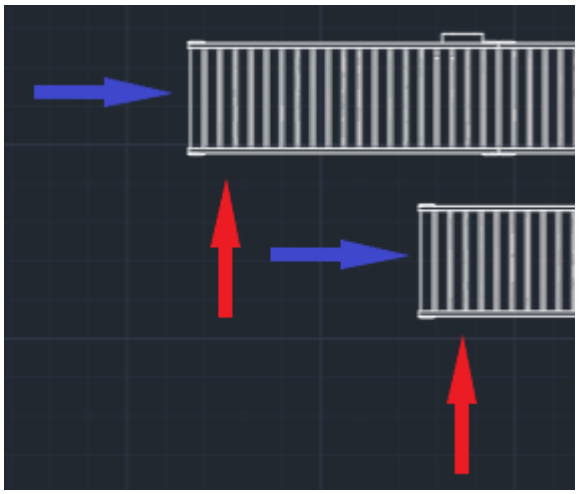
7.2 Esisuunnittelu

Tässä luvussa esitetään ensin yleisesti lajittelulinjaston toimivuus ja sen jälkeen neljä vaihtoehtoista mallia lajittelulinjastolle. Linjasto koostuu nippukuljettimista, optimivalais-

tusta kuvauskopista, roboteista, sitomakoneesta, korvaleikkurista, kuvauspöydästä, kamerasta ja ohjauspaneelista. Liitteessä B on esitetty lajittelulinjaston toiminta vuokaaviona. Ensin nippu tulee linjastolle, jonka jälkeen kaikista katodeista otetaan kuvat molemmilta puolilta. Kuvien perusteella katodit lajitellaan oikeisiin laatuihin ja robotti siirtää katodit oikeisiin nippuihin.

Linjasto voitaisiin sijoittaa esimerkiksi NNH:n leikkaamo rakennukseen, jonne nykyiset niput viedään odottamaan leikkaamista. Ensimmäinen kuljetin voisi alkaa leikkaamo rakennuksen ulkopuolelta, johon nippu tuodaan elektrolyysistä. Näin trukin ei tarvitse tulla sisälle asti, jossa on muutenkin paljon trukkiliikennettä. Lisäksi likaa tulee vähemmän sisälle trukin renkaista, eikä pöly lennä niin paljon. Nippu tuodaan elektrolyysistä ilman teräsvannetta. Kuljetin vie nipun seinän läpi lajittelulinjastolle. Katodin saavuttua optimivalaistuun kuvauskoppiin robotti ja kamera ottavat yhteistyössä tarvittavat kuvat ja robotti vie katodin oikeaan paikkaan lajitellun laadun perusteella.

Jos robotit asennetaan optimivalaistun kuvauskopin sisäpuolelle, sinne on asennettava myös kamerat, jotta kuvauskoppiin nähdään ulkopuolelta. Jotta kuvauskoppiin pääsee mahdollisimman vähän ulkopuolelta valoa, nippujen ja kuljettimien muodostamille aukoille voisi asentaa verhot, jonka läpi niput tulevat. Jokaiselle laatuluokalle on omat kuljettimensa. Kuljettimien pitää olla eri mittaiset ja niiden päässä pitää olla tarpeeksi tilaa, jotta jokaiselta kuljettimelta pääsee hakemaan niput trukilla. Vaihtoehtoisesti niput voidaan noutaa trukilla eri suunnasta, jos trukki mahtuu hakemaan ne sitä kautta. Tällöin kuljettimet voivat olla saman mittaiset. Nippujen mahdolliset noutosuunnat on esitetty Kuvassa 45.



Kuva 45. *Nippujen mahdolliset noutosuunnat.*

Kun nippu saapuu robotin ulottuville, nipun yläpuolella voi olla kamera, joka ottaa kuvan päällimmäisestä katodista. Kun katodista on saatu kuva robotti kääntää sen 180 astetta

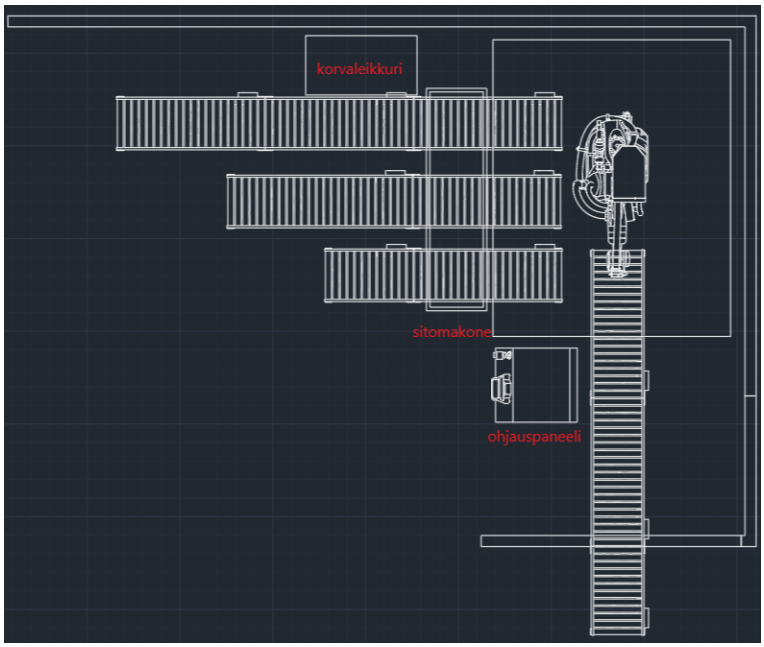
Kuva 46. *Asemointipöytä tyhjänä (vasemmalla) ja katodin kanssa (oikealla).*

ja pitää sitä paikallaan. Näin saadaan otettua molemmilta puolilta kuva. Tällöin saattaa olla ongelmana se, että eri puolen kuvat on otettu eri korkeudelta. Vaihtoehtoisesti kuvauspiste, johon robotti asettaa katodin, voi olla erillinen. Tällöinkin robotin on käännettävä katodi, jotta molemmilta puolilta saadaan kuva. Erillisellä kuvauspisteellä saataisiin kuva vain yhdestä katodista kerrallaan, mutta siihen voisi lisätä vaa'an, jolloin saataisiin myös katodin paino selville. Erillinen kuvauspiste voisi olla saman tyyppinen kuin elektrolyysissä oleva siemenlevyjen asemointipöytä. Tämä on esitetty Kuvassa 46.

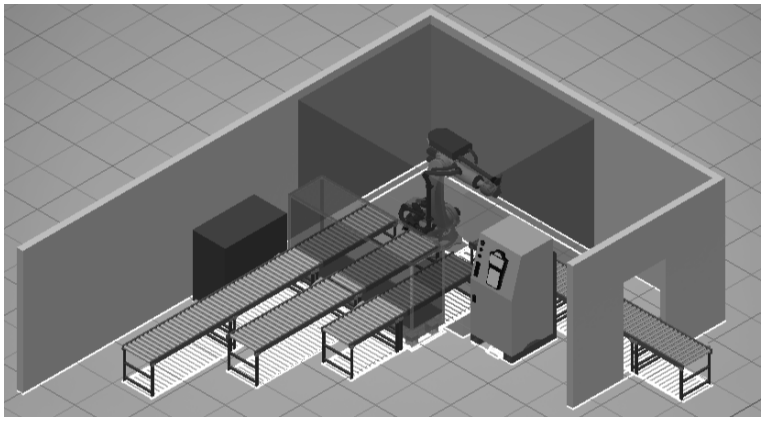
Jokaisessa paikassa, johon katodeja pinotaan, pitää olla vaaka. Nipun paino tarvitaan, jotta tiedetään, milloin nippu tulee valmiiksi. Kun nippu tulee valmiiksi, ohjelma saa siitä tiedon ja antaa kuljettimelle käskyn siirtää nippu eteenpäin. Myös yksittäisen katodin paino tarvitaan, jotta tiedetään, onko kyseessä romukatodi.

Ainakin standardilaadun ja priimalaadun nipuilla pitää olla mahdollisuus sidontaan. Jos mahdollista, kuljettimet olisivat vierekkäin ja niillä olisi yhteinen sitomakone, joka sitoo aina valmiin nipun tarvittaessa. Romuniput valmistuvat hitaammin ja eivät välttämättä tarvitse automaattista sitomista, vaan ne voidaan sitoa manuaalisesti.

Ensimmäisessä mallissa on kolme laatuluokkaa ja jokaisella on oma kuljetin. Kuljettimilla on yhteinen sitomakone, joka sitoo valmiit niput. Leikkaukseen menevistä nipuista leikataan korvat pois leikkauksen yhteydessä, mutta tässä mallissa korvaleikkuri on siirretty lajittelulinjastoon. Malli on esitetty 2D:nä Kuvassa 47 ja 3D:nä Kuvassa 48. Katodin kuva otetaan nipun päältä, jonka jälkeen robotti kääntää katodia 180 astetta, jolloin saadaan



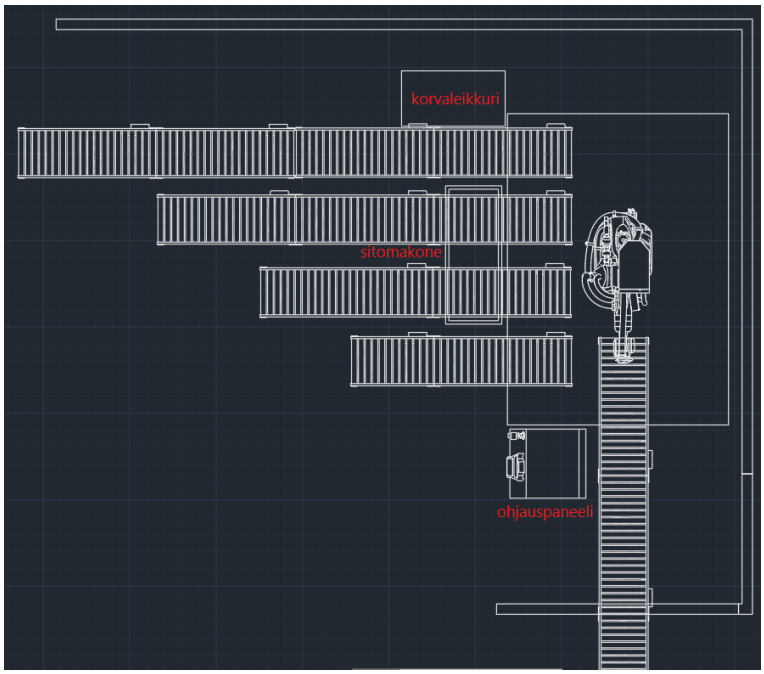
Kuva 47. *Katodien lajittelulinjasto malli 1 2D:nä.*



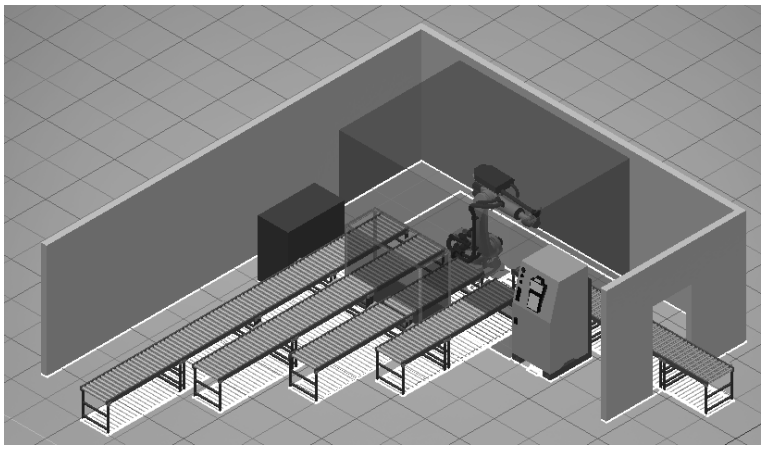
Kuva 48. *Katodien lajittelulinjasto malli 1 3D:nä.*

kuva myös toiselta puolelta. Tässä mallissa myös romuniput kulkevat yhteisen sitomakoneen alta, jolloin niitä ei tarvitsisi sitoa käsin.

Toisessa mallissa laatuluokkia on neljä, joista viimeinen on leikkaukseen menevät niput, joita ei sidota. Näin ollen sitomakone on asetettu vain kahdelle keskimmäiselle kuljettimelle. Neljäs laatu on hieman parempaa laatua kuin standardilaadun katodi, muttei kuitenkaan priimalaatua. Standardilaadun katodeista osa voi olla liian marjaista eli dendriittistä leikattavaksi, vaikka se ei kuitenkaan olisi romua. Malli 2 on esitetty kuvissa 49 ja 50.

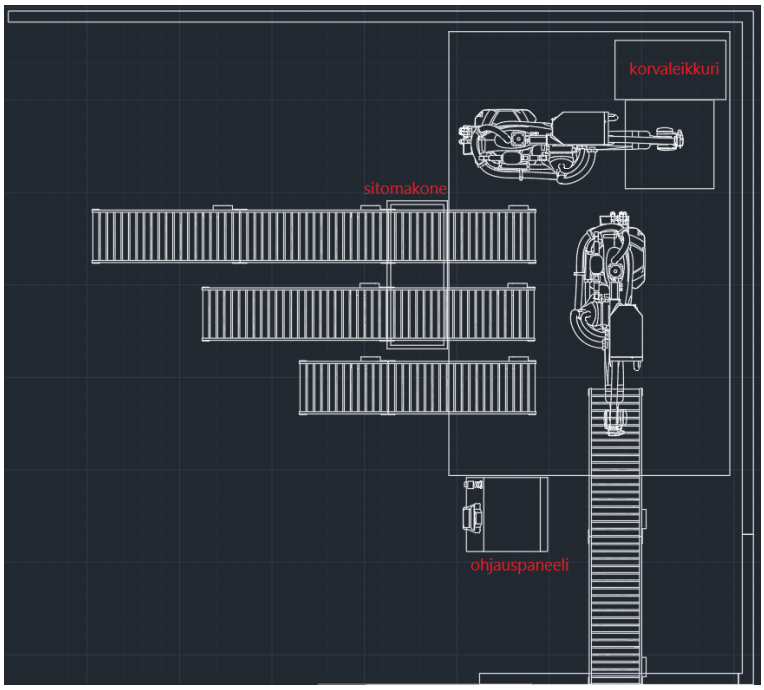


Kuva 49. *Katodien lajittelulinjasto malli 2 2D:nä.*

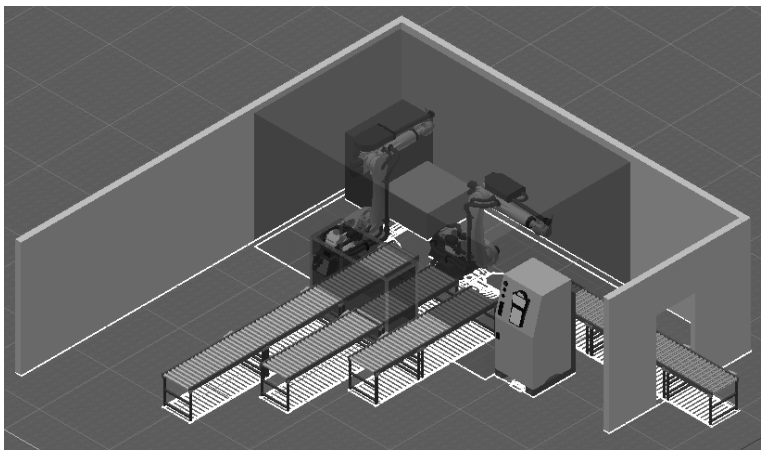


Kuva 50. *Katodien lajittelulinjasto malli 2 3D:nä.*

Kolmannessa mallissa on kaksi robottia. Ensimmäinen robotti kuvaa katodin, jonka jälkeen se vie katodin korvien leikkauspisteelle. Tästä toinen robotti poimii sen korvien leikkauksen jälkeen ja vie oikean laatuluokan kuljettimelle. Malli on esitetty Kuvassa 51 2D:nä ja Kuvassa 52 3D:nä.

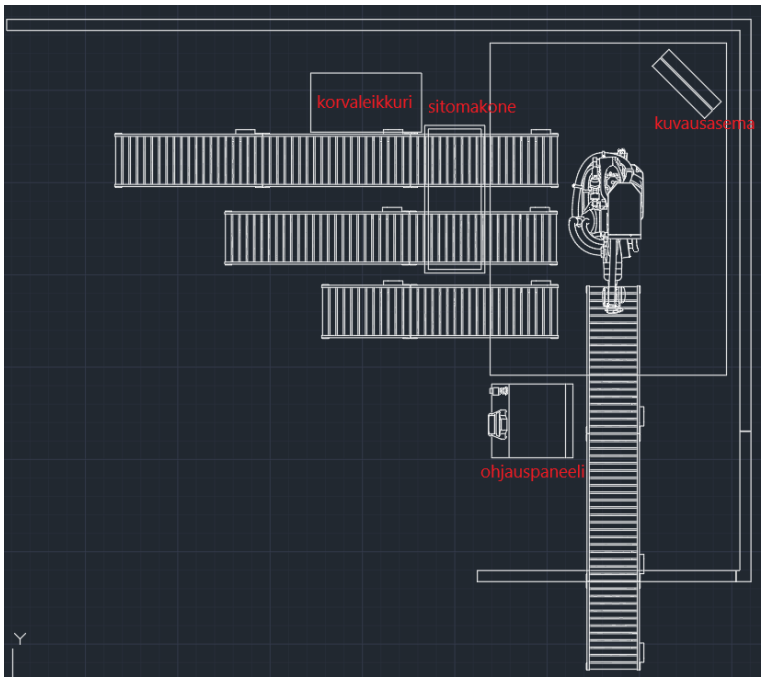


Kuva 51. *Katodien lajittelulinjasto malli 3 2D:nä.*

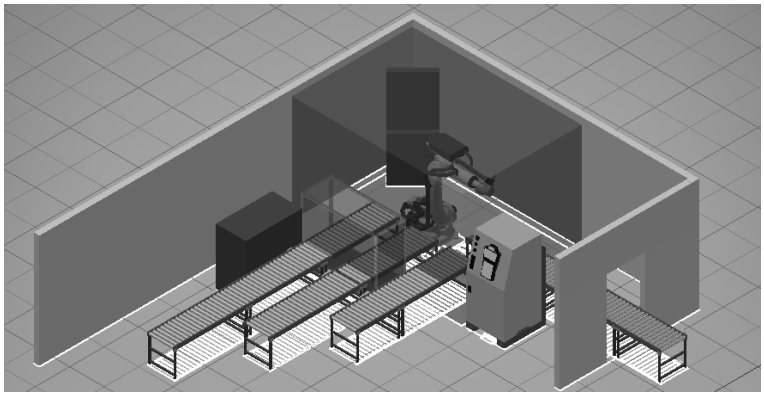


Kuva 52. *Katodien lajittelulinjasto malli 3 3D:nä.*

Neljännessä mallissa robotti asettaa katodin kuvausasemaan, jossa kamera ottaa siitä kuvan. Kamera voi olla kiinteässä paikassa tai robotin päässä. Kun kuva on otettu, robotti kääntää katodin, jotta myös toiselta puolelta saadaan kuva. Kuvausasemassa on myös vaaka katodin punnitusta varten. Tässä mallissa etuna on, että katodit saadaan kuvattua yksittäin. Lisäksi, jos esimerkiksi kamera on robotin päässä, samalla kameralla voitaisiin kuvata katodi sivusta ja päältä, jolloin saataisiin tieto sen käyryydestä. Malli on esitetty Kuvissa 53 ja 54.



Kuva 53. *Katodien lajittelulinjasto malli 4 2D:nä.*



Kuva 54. *Katodien lajittelulinjasto malli 4 3D:nä.*

Edellä olevien mallien ominaisuuksia voi yhdistellä tai ottaa vaikka kaikki ominaisuudet mukaan linjaston suunnitteluun. Esimerkiksi mallin 3 robottien väliin voi asettaa kuvausaseman, jolloin mallin 3 ja 4 ominaisuudet saadaan yhdistettyä.

7.3 Kustannusarvio

Tässä luvussa esitetään arviot linjastolle tarvittavista laitteista sekä karkea arvio niiden kustannuksista. Alueen koko, johon linjastoa on suunniteltu, on noin 16 m x 14 m. Linjastolle tarvitaan optimivalaistu kuvauskoppi, kuljettimia, robotteja, valoverhoja, sitomakone, kuvausasema, katodien nostin ja kameroita.

Kuvauskopin koko riippuu robottien määrästä ja niiden ulottuvuuksista. Sen lattiapinta-ala on arviolta 25-150 m^2. Korkeus on arviolta 5-7 m. Kuvauskoppiin tarvitaan vain seinät

ja katto, jotta ulkopuolelta tuleva valo saadaan eristettyä. Samankokoisen talon rakentaminen Suomessa maksaisi noin 100 000-300 000 € ja Englannissa samankokoiset teollisuuskokonaisuudet 20 000-300 000 € riippuen rakennuksen tyypistä [78, 79]. Tässä kyseessä kuitenkin vain seinät ja katto, joten kustannusarvio on 10 000-200 000 €.

Kuvauskoppiin tarvitaan myös valaistus. Siihen riittää, että koppi saadaan valaistua hyvin. Hallivalaistusten hinta on noin 30- 300 €/kpl [80]. Konenäköön suunnitellut valot ovat noin 100-500 €/kpl [81]. Valoja tarvitsee olla riittävästi ja tarvittava määrä riippuu kopin koosta. Jos tarvittava määrä olisi esimerkiksi 5-20, kustannusarvio olisi 200-6 000 €.

Robotteja linjastolle tarvitaan 1-2 ja niiden hyötykuorman pitää olla yli 100 kg, jotta se on varmasti riittävä. Teollisuusrobottien hinnat ovat 50 000-200 000 € [82, 83]. Lajittelulinjastolle robottien kustannusarvio on 50 000-400 000 €. Roboteille tarvitaan myös oikeanlaiset tarttujat. Tarttujien hinta vaihtelee välillä 10-3 000 € [84, 85]. Lajittelulinjastolle kustannusarvio on 50-10 000 €.

Kuljettimia tarvitaan 30-50 m ja käytettynä samantyyppiset kuljettimet maksavat esimerkiksi alibaba.com ja ebay.com sivuilla noin 1 000-20 000 €. Lajittelulinjaston kuljettimet pitää kuitenkin olla uudet ja kestävämmät, sillä esimerkit näyttivät siltä, etteivät ne kestäisi montaa katodinippua. Kustannusarvio kuljettimille näiden perusteella olisi 10 000-100 000 €.

Kameroita linjastolle tarvitaan katodien kuvaamiseen 1-2 ja lisäksi ainakin yksi yleinen kuvauskopin kamera. Konenäkökameroiden hinnat ovat 200-5 000 €. Linjastolle tarvittavien kameroiden kustannusarvio on 500-15 000 €.

Linjastolle tarvitaan turvalaitteistoja kuten valoverhoja, rajakytkimiä ja kuvauskoppiin turvaovi. Valoverhojen hinnat ovat 500-5 000 €/pari ja niitä tarvitaan ainakin 3 paria [85, 86]. Turva-oven ja kytkimien kustannukset voidaan arvioida samansuuruisiksi, jolloin Turvalaitteiden kustannusarvio on 2 500-30 000 €.

Valmiiksi sopivaa sitomakonetta ei markkinoilta löydy, mutta automaattiset koneet maksavat 1 000-20 000 € [87, 88]. Lajittelulinjastolle tarvitaan linjastolle suunniteltu malli, jolloin kustannusarvio on 10 000-100 000 €.

Edellä mainittujen lisäksi voi olla myös muita kustannuksia esimerkiksi sähköjen asentamisesta sekä työvoimakustannuksista. Kuvausasemaa ja nostimen kustannuksia ei arvioitu ja lisäksi myös konenäköohjelma maksaa myös jotain. Näiden perusteella koko lajittelulinjaston karkea kustannusarvio on 100 000-1 000 000 €, joka kuitenkin on todennäköisesti vielä alakanttiin.

8. YHTEENVETO

Monella alalla on paljon erilaisia lajittelu- ja luokittelutehtäviä. Näitä ovat esimerkiksi tuotteiden hylkäämiset, laadun erottelut, vikojen tunnistukset sekä eri tuotteiden erottelut. Manuaalinen lajittelu on usein huono vaihtoehto lajitteluun ja konenäöstä on monesta tutkimuksesta saatu hyviä tuloksia. Manuaalinen lajittelu voi olla epäjohdonmukaista, aikaa vievää sekä pitkästyttävää työtä. Automaattisella lajittelulla saadaan vapautettua työvoimaa muihin työtehtäviin sekä parannettua lajittelun tuloksia.

Tässä työssä tarkoituksena oli selvittää konenäön soveltuvuus nikkelikatodien lajitteluun sekä suunnitella mahdollista lajittelulinjastoa ja siinä huomioitavia asioita. Tällä hetkellä lajittelua tehdään manuaalisesti laitteistolla, jota ei ole siihen suunniteltu. Lisäksi lajittelu sitoo työvoimaa. Automaattisella lajittelulla saataisiin työstä tehtyä turvallisempaa sekä johdonmukaisempaa.

Lajittelua testattiin Labra.AI:n Vision ohjelmalla, jossa nikkelikatodeja lajiteltiin manuaalisesti imukuppinostimen avulla. Jokaisesta katodista otettiin kuva ja kuvat opetettiin Visionille. Opettamisen jälkeen mallia testattiin. Visionilla päästiin parhaimmillaan 93 % tarkkuuteen ja jos dataa olisi kerätty enemmän, tuloksia olisi todennäköisesti saatu vieläkin paremmiksi. Olisi myös ollut hyvä, jos viimeisenä testinä olisi vielä tehty lopullinen testi, jossa olisi käyty eniten katodeja läpi eli se olisi ollut testijoukoltaan suurin testi.

Visionilla tehtyjen lajittelujen tueksi tehtiin myös ohjelmointitestejä konvoluutioneuroverkoilla avoimen lähdekoodin Tensorflow kirjaston avulla. Näissä testeissä kuvia ei esikäsitelty eikä segmentoitu vaan niissä käytettiin suoraan Visionilla otettuja kuvia. Tarkkuudet olivat parhaimmillaan 74 % ilman esiopetettuja painoja. Esiopetettujen painojen avulla päästiin lopulta 90 %. Jos esikäsittelyä olisi tehty ja dataa olisi ollut enemmän, tuloksia olisi saatettu saada paremmiksi.

Edellä mainittujen testien ja varsinkin Visionilla tehtyjen testien perusteella voidaan todeta, että konenäkö soveltuu hyvin nikkelikatodien lajitteluun. Jatkotutkimuksia ei tarvita ja voidaan todeta, että todennäköisesti saataisiin vielä parempia tarkkuuksia, jos opetettaisiin enemmän nikkelikatodeja. Toimivan linjaston konenäkömallille olisi hyvä opettaa vähintään 500 sekä priimalaadun kuvaa että standardilaadun kuvaa. Jos nikkelikatodeista saadaan yli 90 % luokiteltua oikein, saadaan jo paljon priimalaatua talteen. Priimalaatu muuttuisi todennäköisesti johdonmukaisemmaksi, koska lajittelusta saataisiin ihmisen aiheuttamat laatuvaihtelut pois.

Lajittelulinjaston suunnittelussa annettiin muutama esimerkki erilaisista vaihtoehdoista. Linjasto on suunniteltu sijoitettavaksi NNH:n leikkaamo rakennukseen. Linjasto voisi alkaa rakennuksen ulkopuolelta, jolloin nippuja kuljettavan trukin ei tarvitsisi avata ovia ja ajella sisälle. Lajittelulinjaston kuvaukselle olisi oltava optimivalaistu kuvauskoppi, jotta vältettäisiin valaistuksen muutokset. Jokainen luokka tarvitsee oman paikan, johon niput kasataan. Näiden alla on oltava vaa'at, jotta tiedetään nippujen painot. Lisäksi nippuvaakojen jälkeen voisi olla kuljettimet, jotta nippuja saadaan liikutettua eteenpäin. Kuljettimelle saataisiin myös varastoitua nippuja. Linjastolla olisi hyvä kuvata vain yksi katodi kerrallaan, jotta vältettäisiin päällimmäisen katodin alta pilkottavien katodien vaikutusta luokitteluun. Lisäksi katodin paino vaikuttaa luokitteluun, jolloin voisi hyvä olla erillinen kuvausasema, jossa olisi vaaka ja yksittäinen katodi saataisiin kuvattua.

Linjastolla olisi hyvä olla myös tarvittavien nippujen automaattinen sidonta. Lisäksi mikäli mahdollista, linjastolle tuotavien nippujen olisi hyvä olla kuivia, jotta saataisiin vähennettyä märkien katodien mahdollisesti aiheuttamia ongelmia. Toisaalta, jos kaikki katodit olisivat märkiä, konenäkö voi oppia myös sen hyvin. Linjastolla on oltava manuaalisen luokittelun mahdollisuus esimerkiksi konenäköjärjestelmän ollessa rikki. Konenäköjärjestelmässä olisi hyvä olla mahdollisuus useiden mallien luomiseen sekä lisäopettamiseen.

Linjaston suunnittelussa pitää myös huomioida, jos linjastolle tippuu painavia katodeja, miten ne saadaan sieltä pois. Linjastolla voisi olla katodinosturi, joka olisi tarpeeksi ketterä ja pääsisi mahdollisimman laajalle alueelle, jotta operaattorit välttyvät katodien nostelulta. Tärkeimmät linjaston suunnittelussa huomioitavat asiat on esitetty Taulukossa 9.

Suunnitellulle lajittelulinjastolle tehtiin myös karkea kustannusarvio. Kustannusarviossa huomioitiin linjastolle välttämättömät asiat. Näitä olivat optimivalaistu kuvauskoppi, kuljettimet, robotit, kamerat, valoverhot, valaistus sekä sitomakone. Näille pyrittiin löytämään jonkinlaiset kustannusarviot. Todettiin myös, että linjastolle tarvitaan muitakin asioita, joita ei arvioitu, kuten konenäköohjelma, katodien nostin ja kuvausasema. Kustannusarvioksi saatiin 100 000-1 000 000 €. Arvion uskotaan olevan vielä alakanttiin.

Taulukko 9. *Lajittelulinjaston suunnittelussa huomioitavat asiat.*

Huomioitava asia	**Miksi?**	**Tärkeys**
Molemmilta puolilta kuva	Katodin pitää olla priimaa molemmilta puolilta ollakseen priimaa.	Välttämätön
Yhdestä katodista kuva	Vältetään mahdollisia ongelmia luokittelussa	Tärkeä
Kuvausasema	Saadaan yhdestä katodista kerrallaan kuva ja vaa'an avulla saadaan myös sen paino	Hyödyllinen
Optimivalaistus	Vältetään mahdollisia ongelmia luokittelussa	Välttämätön
Kuivat katodit kuvatessa	Vältetään mahdollisia ongelmia luokittelussa	Hyödyllinen
Kamerojen määrä ja paikka	Kamerat ei saa olla tiellä ja kuvat saatava otettua hyvältä etäisyydeltä, hyvältä paikalta ja nopeasti.	Tärkeä
Katodien käyryys	Käyryys vaikuttaa katodin laatuun	Tärkeä
Robottien määrä, hyötykuorma ja ulottuvuus	Optimoitu prosessi ja robotit soveltuvia tehtävään	Tärkeä
Yhteinen ohjauspaneeli	Yksinkertaisempi käyttää	Hyödyllinen
Manuaalinen luokittelu	Väärät arviot, lisäopetus, uuden mallin opetus	Tärkeä
Kuljettimien määrä	Vaikea lisätä jälkikäteen	Tärkeä
Tarvittavat vaa'at	Yksittäisen katodin sekä valmiin nipun painot tarvitaan	Välttämätön
Nippujen sitominen	Vähentää manuaalista työtä	Tärkeä
Tippuvat katodit	Fyysisen työn minimointi	Tärkeä

LÄHTEET

[1] B. Zhang, W. Huang, J. Li, C. Zhao, S. Fan, J. Wu, C. Liu, Principles, developments and applications of computer vision for external quality inspection of fruits and vegetables: A review, Food Research International, Vol. 62, 2014, pp. 326-343.

[2] N. Neogi, D. Mohanta, P. Dutta, Review of vision-based steel surface inspection systems, EURASIP Journal on Image and Video Processing, Vol. 2014, Iss. 1, 2014, pp. 1-19.

[3] S. Saha, A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way, verkkosivu. Saatavissa (viitattu 13.4.2020): https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[4] Machine learning, verkkosivu. Saatavissa (viitattu 7.4.2020): https://en.wikipedia.org/wiki/Machine_learning.

[5] Nornickel, verkkosivu. Saatavissa (viitattu 30.1.2020): https://www.nornickel.com/.

[6] Harjavallan Suurteollisuuspuisto, verkkosivu. Saatavissa (viitattu 30.1.2020): http://www.suurteollisuuspuisto.com/etusivu.

[7] Norilsk Nickel Harjavalta Oy, verkkosivu. Saatavissa (viitattu 6.10.2020): https://www.kauppalehti.fi/yritykset/yritys/norilsk+nickel+harjavalta+oy/1591728-4.

[8] Nornickel Harjavalta Oy, verkkosivu. Saatavissa (viitattu 30.1.2020): https://www.nornickel.fi/.

[9] T. Laukkanen, kehityspäällikkö, Norilsk Nickel Harjavalta Oy. Haastattelu 12.10.2020.

[10] Labra.AI, verkkosivu. Saatavissa (viitattu 1.9.2020): https://labra.ai/.

[11] B.G. Batchelor, Machine Vision Handbook, 1st ed. Springer London, London, 2012, 3-15 p.

[12] Computer Vision vs. Machine Vision — What's the Difference? verkkosivu. Saatavissa (viitattu 2.6.2020): https://appen.com/blog/computer-vision-vs-machine-vision/.

[13] O. Ghita, T. Carew, P.F. Whelan, A machine vision system for quality grading of painted slates, Springer London, 2012, 1175-1200 p.

[14] Machine vision, verkkosivu. Saatavissa (viitattu 7.4.2020): https://en.wikipedia.org/wiki/Machine_vision.

[15] R. Szeliski, Computer Vision Algorithms and Applications, 1st ed. Springer London, London, 2011, 3-10 p.

[16] Computer vision, verkkosivu. Saatavissa (viitattu 7.4.2020): https://en.wikipedia.org/wiki/Computer_vision.

[17] E. Alpaydin, Introduction to machine learning, MIT Press, Cambridge, Massachusetts, 2014, 1-4 p.

[18] D. Soni, Supervised vs Unsupervised Learning, verkkosivu. Saatavissa (viitattu 7.4.2020): https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[19] Supervised and Unsupervised learning, verkkosivu. Saatavissa (viitattu 7.4.2020): https://www.geeksforgeeks.org/supervised-unsupervised-learning/.

[20] J. Brownlee, A Gentle Introduction to Object Recognition With Deep Learning, verkkosivu. Saatavissa (viitattu 9.10.2020): https://machinelearningmastery.com/object-recognition-with-deep-learning/.

[21] You Only Look Once: Unified, Real-Time Object Detection, IEEE, 2016, pp. 779-788.

[22] Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, IEEE, 2014, pp. 580-587.

[23] D. Parthasarathy, A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN, verkkosivu. Saatavissa (viitattu 12.4.2020): https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4.

[24] R. Girshick, Fast R-CNN, arXiv.org, 2015, https://arxiv.org/abs/1504.08083.

[25] Detection, verkkosivu. Saatavissa (viitattu 12.4.2020): https://www.slideshare.net/simplyin-simple/detection-52781995.

[26] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 39, Iss. 6, 2017, pp. 1137-1149.

[27] K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PP, Iss. 99, 2018, pp. 1.

[28] Otsu Thresholding, verkkosivu. Saatavissa (viitattu 8.4.2020): http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html.

[29] T.Y. Goh, S.N. Basah, H. Yazid, M.J. Aziz Safar, F.S. Ahmad Saad, Performance analysis of image thresholding: Otsu technique, Measurement, Vol. 114, 2018, pp. 298-307.

[30] Background modelling and background subtraction performance for object detection, IEEE, 2010, pp. 1-6.

[31] Robust techniques for background subtraction in urban traffic video, SPIE, 2004, pp. 881-892.

[32] How to Use Background Substraction Methods, verkkosivu. Saatavissa (viitattu 8.4.2020): https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html.

[33] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015, https://arxiv.org/abs/1505.04597.

[34] A. Gandhi, Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2, verkkosivu. Saatavissa (viitattu 26.9.2020): https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/.

[35] H. Sankesara, UNet, verkkosivu. Saatavissa (viitattu 2.10.2020): https://towardsdatascience.com/u-net-b229b32b4a71.

[36] A.E. Orhan, X. Pitkow, Skip Connections Eliminate Singularities, 2017, https://arxiv.org/abs/1701.09175.

[37] V. Powell, L. Lehe, Principal Component Analysis - explained visually, verkkosivu. Saatavissa (viitattu 2.10.2020): https://setosa.io/ev/principal-component-analysis/.

[38] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdisciplinary Reviews: Computational Statistics, Vol. 2, Iss. 4, 2010, pp. 433-459.

[39] Random Cut Forest (RCF) Algorithm, verkkosivu. Saatavissa (viitattu 13.4.2020): https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html.

[40] Sudipto Guha University of Pennsylvania, Philadelphia, PA 19104., Robust Random Cut Forest Based Anomaly Detection On Streams, Vol. 48, 2016, https://www.amazon.science/publications/robust-random-cut-forest-based-anomaly-detection-on-streams.

[41] D. Hudgeon, R. Nichol, Machine Learning for Business, Manning Publications, 2020, Chapter 5.6.

[42] O. Harrison, Machine Learning Basics with the K-Nearest Neighbors Algorithm, verkkosivu. Saatavissa (viitattu 13.4.2020): https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[43] F. Li, R. Krishna & D. Xu, Lecture 2: Image Classification A Core Task in Computer Vision, verkkosivu. Saatavissa (viitattu 2.10.2020): http://cs231n.stanford.edu/slides/2020/lecture_2.pdf.

[44] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT Press, Cambridge, MA, 2017, 326-366 p.

[45] A. Shamsaldin, P. Fattah, T. Rashid, N. Al-Salihi, The Study of The Convolutional Neural Networks Applications, UKH Journal of Science and Engineering, Vol. 3, Iss. 2, 2019, pp. 31-40.

[46] A. Deshpande, A Beginner's Guide To Understanding Convolutional Neural Networks Part 2, verkkosivu. Saatavissa (viitattu 24.9.2020): https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/.

[47] H. El-Amir, M. Hamdy, Deep learning pipeline: building a deep learning model with TensorFlow, Apress, Place of publication not identified, 2020, Chapter 11.

[48] Prabhu, Understanding of Convolutional Neural Network (CNN) — Deep Learning, verkkosivu. Saatavissa (viitattu 23.9.2020): https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[49] J. Brownlee, Difference Between a Batch and an Epoch in a Neural Network, verkkosivu. Saatavissa (viitattu 23.9.2020): https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/.

[50] A. Mohanty, Multi layer Perceptron (MLP) Models on Real World Banking Data, verkkosivu. Saatavissa (viitattu 30.9.2020): https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f.

[51] CNN Training With Code Example - Neural Network Programming Course, verkkosivu. Saatavissa (viitattu 11.10.2020): https://deeplizard.com/learn/video/0VCOG8IeVf8.

[52] J. Brownlee, How to Avoid Overfitting in Deep Learning Neural Networks, verkkosivu. Saatavissa (viitattu 11.10.2020): https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/.

[53] R. Ruizendaal, Deep Learning #3: More on CNNs & Handling Overfitting, verkkosivu. Saatavissa (viitattu 11.10.2020): https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d.

[54] Gaussian blur, verkkosivu. Saatavissa (viitattu 30.9.2020): https://en.wikipedia.org/wiki/Gaussian_blur.

[55] Labra.AI. Haastattelu 29.6.2020.

[56] E.R. Davies, The application of machine vision to food and agriculture: a review, The Imaging Science Journal, Vol. 57, Iss. 4, 2009, pp. 197-217.

[57] D. Sanchez, S. Bulon, L. Moreno, A. Birlutiu, M. Kadar, AUTOMATIC CHARACTER RECOGNITION IN PORCELAIN WARE, Acta Technica Napocensis, Vol. 59, Iss. 3, 2018, pp. 8-12.

[58] J. Silveira, M. Ferreira, C. Santos, T. Martins, Computer Vision Techniques Applied to the Quality Control of Ceramic Plates, 2009.

[59] Deep learning implementation using convolutional neural network in mangosteen surface defect detection, in: ICCSCE, IEEE, 2017, pp. 242-246.

[60] N. Teimouri, M. Omid, K. Mollazade, A. Rajabipour, N. Teimouri, An Artificial Neural Network-Based Method to Identify Five Classes of Almond According to Visual Features, Journal of Food Process Engineering, Vol. 39, Iss. 6, 2016, pp. 625-635.

[61] S. Koyanaka, K. Kobayashi, Automatic sorting of lightweight metal scrap by sensing apparent density and three-dimensional shape, Resources Conservation And Recycling; Resour.Conserv.Recycl., Vol. 54, Iss. 9, 2010, pp. 571-578.

[62] S. Koyanaka, K. Kobayashi, Incorporation of neural network analysis into a technique for automatically sorting lightweight metal scrap generated by ELV shredder facilities, Resources, Conservation & Recycling, Vol. 55, Iss. 5, 2011, pp. 515-523.

[63] K. Nakano, Application of neural networks to the color grading of apples, Computers and Electronics in Agriculture, Vol. 18, Iss. 2, 1997, pp. 105-116.

[64] G. ElMasry, S. Cubero, E. Moltó, J. Blasco, In-line sorting of irregular potatoes by using automated computer-based machine vision system, Journal of Food Engineering, Vol. 112, Iss. 1-2, 2012, pp. 60-68.

[65] Defect Detection in Porcelain Industry Based on Deep Learning Techniques, IEEE, 2017, pp. 263-270.

[66] B. Park, Y.R. Chen, M. Nguyen, Multi-spectral Image Analysis using Neural Network Algorithm for Inspection of Poultry Carcasses, Journal of Agricultural Engineering Research, Vol. 69, Iss. 4, 1998, pp. 351-363.

[67] A computer vision system for automatic steel surface inspection, IEEE, 2010, pp. 1667-1670.

[68] K. Sabanci, A. Kayabasi, A. Toktas, Computer vision-based method for classification of wheat grains using artificial neural network: Computer vision-based method for classification, Journal of the science of food and agriculture, Vol. 97, Iss. 8, 2017, pp. 2588-2593.

[69] Tensorflow, verkkosivu. Saatavissa (viitattu 18.9.2020): https://www.tensorflow.org/.

[70] Keras, verkkosivu. Saatavissa (viitattu 18.9.2020): https://keras.io/about/.

[71] OpenCV, verkkosivu. Saatavissa (viitattu 30.9.2020): https://opencv.org/about/.

[72] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, Vol. 86, Iss. 11, 1998, pp. 2278-2324.

[73] M. Rizwan, LeNet-5 – A Classic CNN Architecture, verkkosivu. Saatavissa (viitattu 10.10.2020): https://engmrk.com/lenet-5-a-classic-cnn-architecture/.

[74] V. Roman, CNN Transfer Learning & Fine Tuning, verkkosivu. Saatavissa (viitattu 8.10.2020): https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2.

[75] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017, https://arxiv.org/abs/1704.04861.

[76] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, 2016, pp. 770-778. https://arxiv.org/abs/1512.03385.

[77] The CIFAR-10 dataset, verkkosivu. Saatavissa (viitattu 9.10.2020): https://www.cs.toronto.edu/~kriz/cifar.html.

[78] Pikakustannuslaskuri, verkkosivu. Saatavissa (viitattu 13.10.2020): https://www.rakentaja.fi/kustannusarvio/.

[79] TYPICAL UK CONSTRUCTION COSTS OF BUILDINGS, verkkosivu. Saatavissa (viitattu 13.10.2020): https://costmodelling.com/building-costs.

[80] Teollisuus- ja hallivalaistus, verkkosivu. Saatavissa (viitattu 13.10.2020): https://www.led-valot.fi/teollisuus-ja-hallivalaistus?dir=asc&order=price.

[81] Industrial machine vision camera, verkkosivu. Saatavissa (viitattu 13.10.2020): https://www.get-cameras.com/.

[82] How Much Do Industrial Robots Cost? verkkosivu. Saatavissa (viitattu 13.10.2020): https://sp-automation.co.uk/how-much-do-industrial-robots-cost-3/.

[83] RobotWorkx - How Much Do Industrial Robots Cost? verkkosivu. Saatavissa (viitattu 13.10.2020): https://www.robots.com/faq/how-much-do-industrial-robots-cost.

[84] S. Bouchard, Robot gripper: How much does it cost? verkkosivu. Saatavissa (viitattu 14.10.2020): https://blog.robotiq.com/bid/58812/Robot-gripper-How-much-does-it-cost.

[85] RS Components, verkkosivu. Saatavissa (viitattu 13.10.2020): https://uk.rs-online.com/web/.

[86] Light Curtains - Grainger Industrial Supply, verkkosivu. Saatavissa (viitattu 13.10.2020): https://www.grainger.com/category/electrical/industrial-controls-automation-and-machine-safety/machine-safety-matting-and-switching/machine-safety-switching/light-curtains.

[87] Sareskoski - Turvaa ja tehoa, verkkosivu. Saatavissa (viitattu 13.10.2020): https://www.sareskoski.com/vannekone-poytamalli-puoliautomaatti/P234.

[88] D. Roberge, How Much Does A Strapping Machine Cost? verkkosivu. Saatavissa (viitattu 13.10.2020): https://www.industrialpackaging.com/blog/strapping-machine-cost.

LIITE A: VISIONIN TESTITULOKSET

Opetetut levyt	9	1	romu	monta testattiin	malli		9	1	romu		Kokonais	
(9+muut)												
30+40	2/6			1 nippu			0,33					
50+100	0/10			5 nippua	valo ja pimee kuvat sekasin							
41+72	4/7		3 väärää	2 nippua	pimeä		0,57		0,93		**0,88**	
71+104	13/18		7 väärää	3 nippua	pimeä		0,72		0,87		**0,79**	
104+141	19/22		6 väärää	3 nippua	pimeä		0,86		0,88		**0,88**	
180+180	59/72		8/20	4 nippua	pimeä ja uudet valosat		0,82		0,4		**0,73**	
198+198	44/58		37/41	5 nippua	pimeä ja lisää valosia		0,76		0,9		**0,82**	
(9+std+romu)												
155+104+9	1/1	2/22	2/2	1 nippu	valoisa		1	0,09	1		**0,2**	Testi 1
169+170+40	45/63	84/98	0/0	8 nippua		normi testi	0,71	0,86	1		**0,8**	
-II-	12/13	29/37	25/32	4 nippua		romuniput+1 normi	0,92	0,78	0,78		**0,8**	
yhteensä	57/76	113/135	25/32	12 nippua	valoisa	kaksi edellistä	0,84	0,84	0,78		**0,8**	Testi 2
220+232+71	71/74	46/57	19/21	6 nippua	valoisa		0,96	0,81	0,9		**0,89**	Testi 3
223+243+82	37/44	86/93	33/38	8 nippua	valoisa	5 romunip, param	0,84	0,92	0,87		**0,89**	Testi 4
250+250+114	45/61	50/63	8/9	7 nippua	valoisa	1 rnip, nosturi ja valo	0,74	0,79	0,89		**0,78**	Testi 5
272+265+113	78/81	37/43	10/10	6 nippua	valoisa	1 rnip	0,96	0,86	1		**0,93**	Testi 6
uus päivitys	augmentointi testit											
270+282+118	26/29	4/9	1/1	2 nippua		simple	0,89	0,44	1		**0,79**	Testi 7
270+282+118	33/36	4/6	1/1	2 nippua		oma	0,92	0,67	1		**0,88**	Testi 8
270+284+120	98/113	30/34	4/4	7 nippua		triplatreeni	0,87	0,88	1		**0,87**	Testi 9
280+294+121	113/129	89/98	2/2	10 nippua		välitreenit	0,88	0,91	1		**0,89**	
280+294+121	15/19	21/24	0/0	2 nippua		vielä vanhalla	0,79	0,88	1		**0,84**	
yhteensä	128/148	110/122	2/2	12 nippua		kaksi edellistä	0,86	0,9	1		**0,88**	Testi 10
300+310+124	28/28	0/4	0/0	1,5 nippua		triplatreeni (ei tunnistanut 1)	1	0	1			
300+314+124	59/64	36/40	0/0	4,5 nippua		uus treeni	0,92	0,9	1		**0,91**	Testi 11

Treeni:	val loss	val status	val obj acc	train statu	train obj acc
Testi 4	0,39	1	0,87	0,99	0,99
Testi 5	0,21	1	0,94	0,98	0,98
Testi 6	0,26	1	0,9	0,98	0,98
Testi 7	0,7		0,6		0,91
Testi 8	0,68		0,61		0,91
Testi 9	0,68		0,66		0,91
Testi 10	0,69		0,69		0,86
	0,72		0,65		0,9
Testi 11	0,79		0,63		0,9

Väärin menneet:	Testi 4	Testi 5	Testi 6	Testi 7	Testi 8	Testi 9	Testi 10	Testi 11
9, mutta sanoo 1	7	13	3	3	3	15	20	5
9, mutta sanoo romu	0	3	0	0	0	0	0	0
1, mutta sanoo 9	6	5	6	5	2	4	12	4
1, mutta sanoo romu	1	8	0	0	0	0	0	0
romu, mutta sanoo 9	1	0	0	0	0	0	0	0
romu, mutta sanoo 1	4	1	0	0	0	0	0	0
virheitä yhteensä	21/175	30/133	9/134	8/39	5/43	19/151	32/270	9/104

LIITE B: LAJITELULINJASTON VUOKAAVIO